S32R274 Series Reference Manual

Document Number: S32R274 Reference Manual
Rev. 4, 05/2018

Contents

Section number **Title** **Page**

Chapter 1
About This Manual

Chapter 2
Introduction

Chapter 3
Memory Map

Chapter 4
Signal Multiplexing and Signal Description

Chapter 5
Clocking Overview

Chapter 6
Reset Overview

Chapter 7
System Boot

Chapter 8
Interrupt and DMA Overview

Chapter 9
Functional Safety

Chapter 10
Embedded memories

Chapter 11
Device Configuration Format (DCF) Records

Chapter 12
System Integration Unit Lite2 (SIUL2)

Chapter 13
Core Complex Overview

Chapter 14
Core description

Chapter 15
z7 Core Description

Chapter 16
Nexus Module

Chapter 17
z7 Nexus Module

Chapter 18
e200z4201n3 Core Debug Support

Chapter 19

z7 Core Debug Support

Chapter 20
Signal Processing Extension (SPE)

Chapter 21
Crossbar Switch (AXBS)

Chapter 22
Crossbar Integrity Checker (XBIC)

Chapter 23
Peripheral Bridge (AIPS-Lite)

Chapter 24
System Memory Protection Unit (SMPU)

Chapter 25
Semaphores2 (SEMA42)

Chapter 26
Development Trigger Semaphore (DTS)

Chapter 27
Wakeup Unit (WKPU)

Chapter 28
Interrupt Controller (INTC)

Chapter 29
Direct Memory Access Multiplexer (DMAMUX)

Chapter 30
Platform Configuration Module (PCM)

Chapter 31
Multipurpose Control Block (MCB)

Chapter 32
Dual PLL Digital Interface (PLLDIG)

Chapter 33
Clock Monitor Unit (CMU)

Chapter 34
Clock Generation Module (MC_CGM)

Chapter 35
IRCOSC Digital Interface

Chapter 36
Enhanced Direct Memory Access (eDMA)

Chapter 37
Quad-ported RAM controller (PRAMC)

Chapter 38
Flash memory controller

Chapter 39
Embedded Flash Memory (c55fmc)

Chapter 40
Cross-Triggering Unit (CTU)

Chapter 41
Enhanced Motor Control Timer (eTimer)

Chapter 42

Motor Control Pulse Width Modulator (FlexPWM)

Chapter 43
Analog-to-Digital Converter (ADC)

Chapter 44
Temperature Sensor (TSENS)

Chapter 45
Signal Processing Toolbox (SPT)

Chapter 46
Cross Triggering Engine (CTE)

Chapter 47
Waveform Generation Module (WGM)

Chapter 48
RADAR Analog Front-End (AFE)

Chapter 49
MIPICSI2

Chapter 50
System Timer Module (STM)

Chapter 51
Software Watchdog Timer (SWT)

Chapter 52
Periodic Interrupt Timer (PIT)

Chapter 53
CAN (FlexCAN)

Chapter 54
Zipwire

Chapter 55
Serial Interprocessor Interface (SIPI)

Chapter 56
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

Chapter 57
Serial Peripheral Interface (SPI)

Chapter 58
LINFlexD

Chapter 59
Inter-Integrated Circuit (I2C)

Chapter 60
FlexRay Communication Controller (FlexRay)

Chapter 61
Ethernet MAC (ENET)

Chapter 62
Reset Generation Module (MC_RGM)

Chapter 63
System Status and Configuration Module (SSCM)

Chapter 64

Power Management Controller block (PMC)

Chapter 65
Power Control Unit (MC_PCU)

Chapter 66
Mode Entry Module (MC_ME)

Chapter 67
Nexus Crossbar Multi-Master Client (NXMC)

Chapter 68
Nexus Port Controller (NPC)

Chapter 69
Nexus Aurora Link (NAL)

Chapter 70
Nexus Aurora PHY (NAP)

Chapter 71
JTAG Controller (JTAGC)

Chapter 72
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

Chapter 73
JTAG Master (JTAGM)

Chapter 74
Cyclic Redundancy Check (CRC) Unit

Chapter 75
Memory Error Management Unit (MEMU)

Chapter 76
Fault Collection and Control Unit (FCCU)

Chapter 77
Self-Test Control Unit (STCU2)

Chapter 78
Error Injection Module (EIM)

Chapter 79
Register Protection (REG_PROT)

Chapter 80
Password and Device Security Module (PASS)

Chapter 81
Tamper Detection Module (TDM)

Chapter 82
Cryptographic Services Engine (CSE)

Chapter 1
About This Manual

1.1 Audience

This reference manual is intended for system software and hardware developers and applications programmers who want to develop products with this device. It assumes that the reader understands operating systems, microprocessor system design, and basic principles of software and hardware.

1.2 Organization

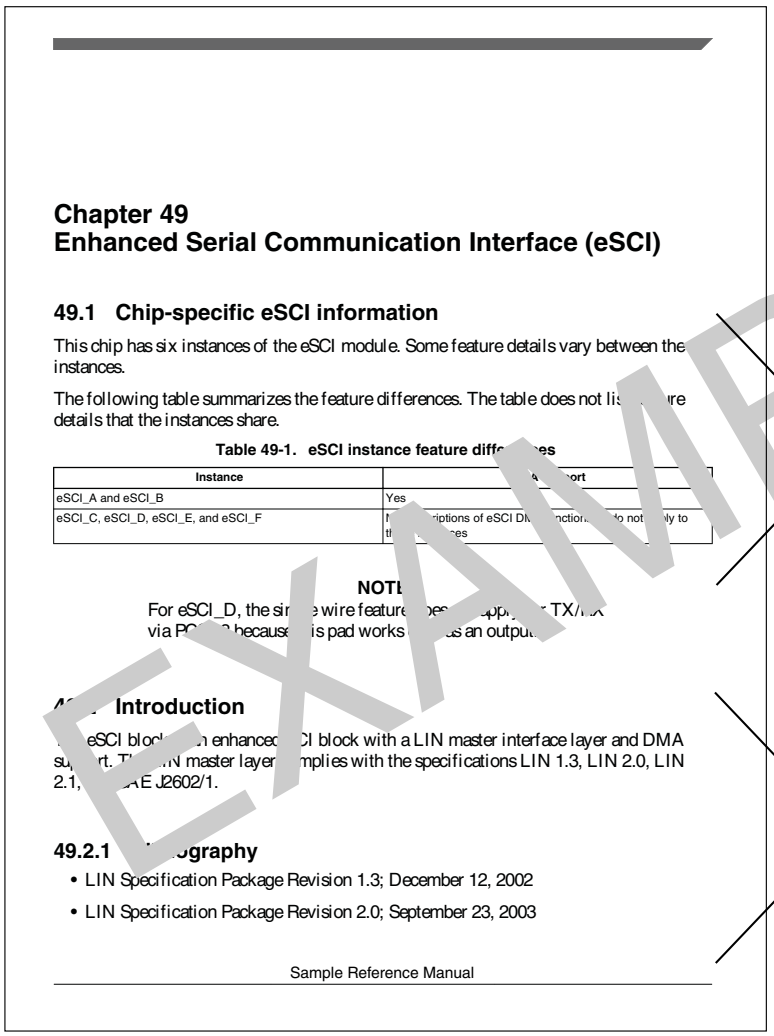
This manual has two main sets of chapters.

1. Chapters in the first set contain information that applies to all components on the chip.
2. Chapters in the second set are organized into functional groupings that detail particular areas of functionality.
   - Examples of these groupings are clocking, timers, and communication interfaces.
   - Each grouping includes chapters that provide a technical description of individual modules.

1.3 Module descriptions

Each module chapter has two main parts:

- The first section, *Chip-specific [module name] information*, provides details such as the number of module instances on the chip and connections between the module and other modules. Read this section *first* because its content is crucial to understanding the information in other sections of the chapter.
- The subsequent sections provide general information about the module, including its signals, registers, and functional description.



Chapter 49
Enhanced Serial Communication Interface (eSCI)

49.1 Chip-specific eSCI information

This chip has six instances of the eSCI module. Some feature details vary between the instances.

The following table summarizes the feature differences. The table does not list feature details that the instances share.

Table 49-1. eSCI instance feature differences

Instance	
eSCI_A and eSCI_B	Yes
eSCI_C, eSCI_D, eSCI_E, and eSCI_F	

NOTE

For eSCI_D, the single wire feature does not support TX/RX via PC because this pad works as an output

Introduction

The eSCI block is an enhanced SCI block with a LIN master interface layer and DMA support. The LIN master layer complies with the specifications LIN 1.3, LIN 2.0, LIN 2.1, and SAE J2602/1.

49.2.1 Bibliography

- LIN Specification Package Revision 1.3; December 12, 2002
- LIN Specification Package Revision 2.0; September 23, 2003

Sample Reference Manual

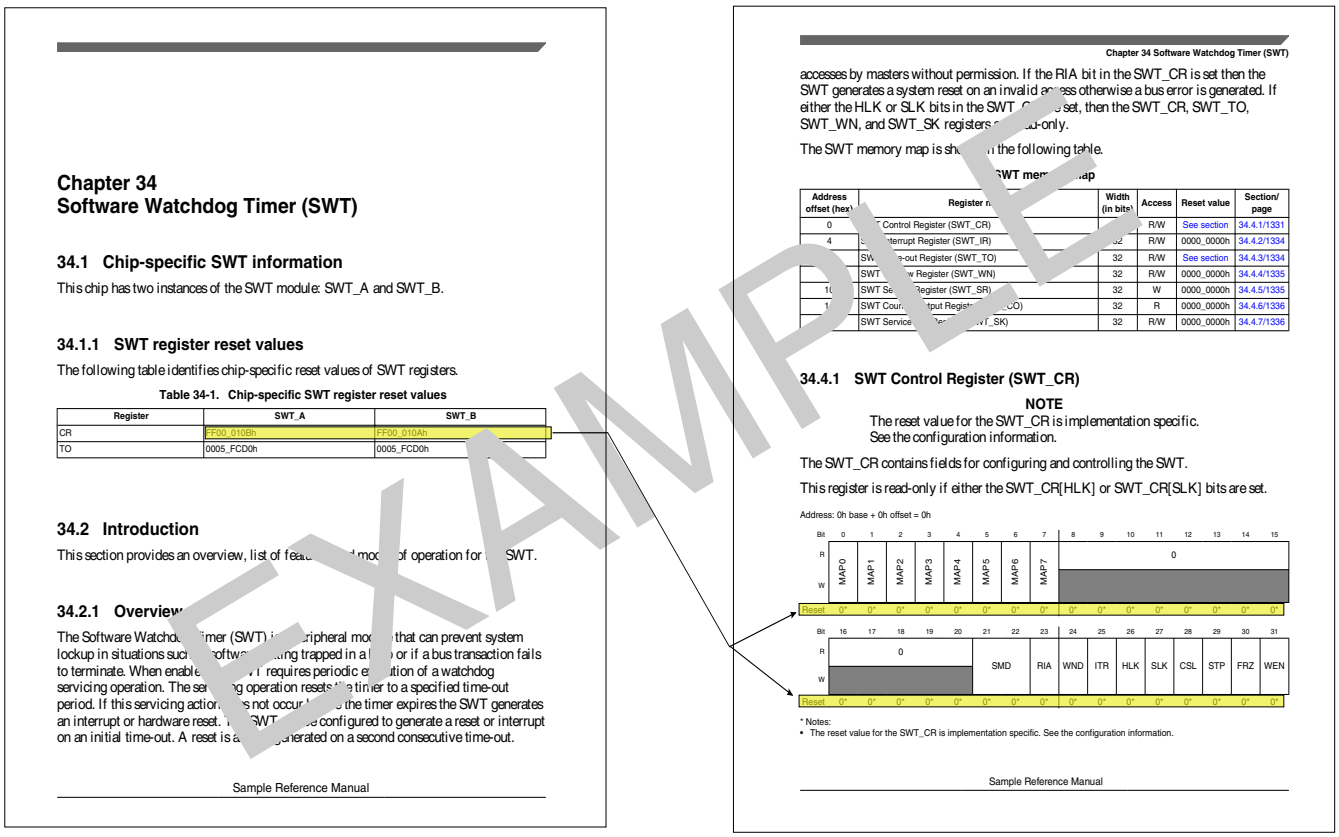
Chip-specific information that should be read first

Beginning of general module information

Figure 1-1. Example: chapter chip-specific information and general module information

1.3.1 Example: chip-specific information that clarifies content in the same chapter

The example below shows chip-specific information that clarifies general module information presented later in the chapter. In this case, the chip-specific register reset values supercede the reset values that appear in the register diagram.



Chapter 34
Software Watchdog Timer (SWT)

34.1 Chip-specific SWT information

This chip has two instances of the SWT module: SWT_A and SWT_B.

34.1.1 SWT register reset values

The following table identifies chip-specific reset values of SWT registers.

Table 34-1. Chip-specific SWT register reset values

Register	SWT_A	SWT_B
CR	FF00_010Bh	FF00_010Ah
TO	0005_FCD0h	0005_FCD0h

34.2 Introduction

This section provides an overview, list of feat... mod... of operation for ... SWT.

34.2.1 Overview

The Software Watchdo... imer (SWT) i... ripheral mo... that can prevent system lockup in situations suc... softw... ting trapped in a l... or if a bus transaction fails to terminate. When enable... T requires periodic e... ution of a watchdog servicing operation. The se... g operation resets ... timer to a specified time-out period. If this servicing actio... s not occur ... the timer expires the SWT generates an interrupt or hardware reset. ... SWT ... e configured to generate a reset or interrupt on an initial time-out. A reset is a... generated on a second consecutive time-out.

Sample Reference Manual

Chapter 34 Software Watchdog Timer (SWT)

accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid a... ss otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_C... set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers ... -only.

The SWT memory map is sh... n the following table.

SWT mem... ap

Address offset (hex)	Register n...	Width (in bits)	Access	Reset value	Section/ page
0	... Control Register (SWT_CR)		R/W	See section	34.4.1/1331
4	... terrupt Register (SWT_IR)	32	R/W	0000_0000h	34.4.2/1334
	SW... e-out Register (SWT_TO)	32	R/W	See section	34.4.3/1334
	SWT ... w Register (SWT_WN)	32	R/W	0000_0000h	34.4.4/1335
10	SWT Se... Register (SWT_SR)	32	W	0000_0000h	34.4.5/1335
1	SWT Cour... tput Regist... CO)	32	R	0000_0000h	34.4.6/1336
	SWT Service ... (SWT_SK)	32	R/W	0000_0000h	34.4.7/1336

34.4.1 SWT Control Register (SWT_CR)

NOTE
The reset value for the SWT_CR is implementation specific.
See the configuration information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD	RIA	WND	ITR	HLK	SLK	CSL	STP	FRZ	WEN	
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

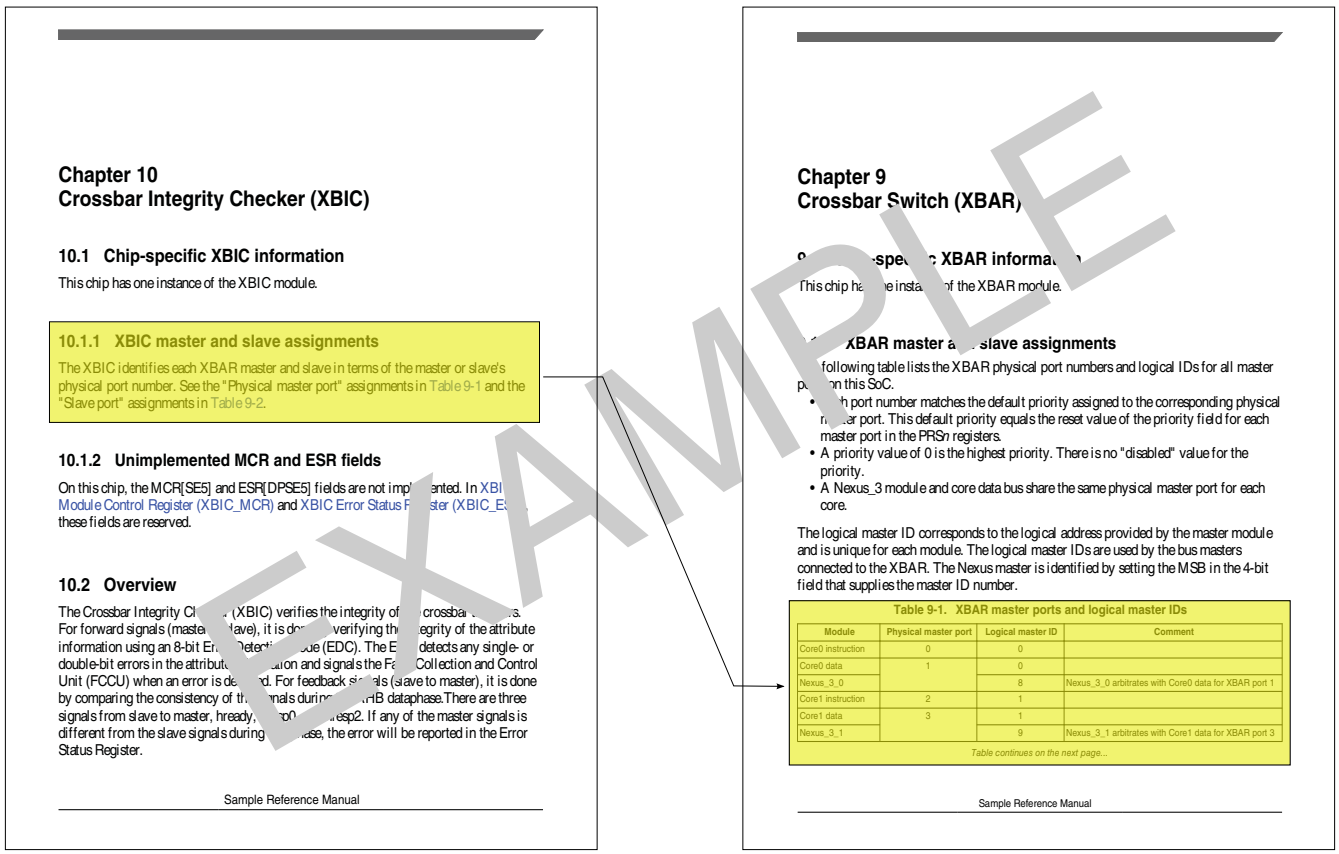
* Notes:
- The reset value for the SWT_CR is implementation specific. See the configuration information.

Sample Reference Manual

Figure 1-2. Example: chip-specific information that clarifies content in the same chapter

1.3.2 Example: chip-specific information that refers to a different chapter

The chip-specific information below refers to another chapter's chip-specific information. In this case, read both sets of chip-specific information before reading further in the chapter.



Chapter 10
Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

The XBIC identifies each XBAR master and slave in terms of the master or slave's physical port number. See the "Physical master port" assignments in Table 9-1 and the "Slave port" assignments in Table 9-2.

10.1.2 Unimplemented MCR and ESR fields

On this chip, the MCR[SE5] and ESR[DPSE5] fields are not implemented. In XBIC Module Control Register (XBIC_MCR) and XBIC Error Status Register (XBIC_ES... these fields are reserved.

10.2 Overview

The Crossbar Integrity Ch... (XBIC) verifies the integrity of ... crossbar ... For forward signals (maste... lave), it is do... verifying th... egrity of the attribute information using an 8-bit Er... Detecti... ue (EDC). The E... detects any single- or double-bit errors in the attribut... tion and signals the Fa... Collection and Control Unit (FCCU) when an error is de... d. For feedback si...ls (slave to master), it is done by comparing the consistency of th...nals during ...hB dataphase.There are three signals from slave to master, hready, ...sp0, ...esp2. If any of the master signals is different from the slave signals during ...ase, the error will be reported in the Error Status Register.

Sample Reference Manual

Chapter 9
Crossbar Switch (XBAR)

9 ...-spe... : XBAR informa...

This chip ha... e insta... of the XBAR module.

... XBAR master a... slave assignments

The following table lists the XBAR physical port numbers and logical IDs for all master p... on this SoC.

- ...h port number matches the default priority assigned to the corresponding physical ...er port. This default priority equals the reset value of the priority field for each master port in the PRSn registers.
- A priority value of 0 is the highest priority. There is no "disabled" value for the priority.
- A Nexus_3 module and core data bus share the same physical master port for each core.

The logical master ID corresponds to the logical address provided by the master module and is unique for each module. The logical master IDs are used by the bus masters connected to the XBAR. The Nexus master is identified by setting the MSB in the 4-bit field that supplies the master ID number.

Table 9-1. XBAR master ports and logical master IDs

Module	Physical master port	Logical master ID	Comment
Core0 instruction	0	0	
Core0 data	1	0	
Nexus_3_0		8	Nexus_3_0 arbitrates with Core0 data for XBAR port 1
Core1 instruction	2	1	
Core1 data	3	1	
Nexus_3_1		9	Nexus_3_1 arbitrates with Core1 data for XBAR port 3

Table continues on the next page...

Sample Reference Manual

Figure 1-3. Example: chip-specific information that refers to a different chapter

1.4 Register descriptions

Module chapters present register information in:

- Memory maps containing:
  - An offset from the module's base address
  - The name and acronym/abbreviation of each register
  - The width of each register (in bits)
  - Each register's reset value
  - The page number on which each register is described
- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in the following figure.

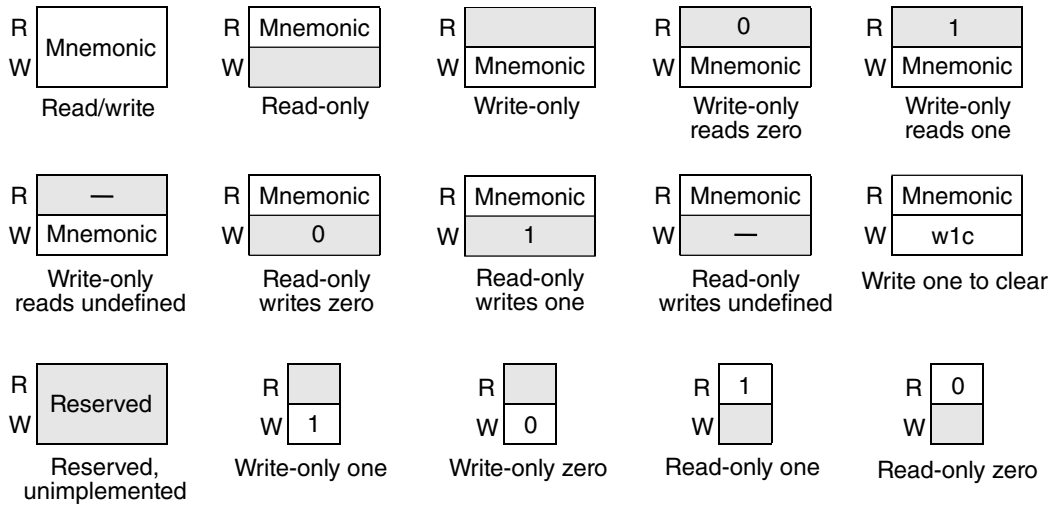


Figure 1-4. Register figure conventions

1.5 Conventions

1.5.1 Notes, Cautions, and Warnings

Note, Caution, and Warning notices appear throughout this manual. Each notice type alerts readers to a specific kind of information.

NOTE

Notes convey information that may be tangential to a topic or that may not apply to all readers.

CAUTION

Caution notices call out information that readers should know before taking further action. Cautions frequently point to trouble spots that may damage the chip or board.

WARNING

Warning notices inform readers about actions that could result in unwanted consequences, especially those that may cause bodily injury.

1.5.2 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*.
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*.

1.5.3 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
placeholder, x	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
`code`	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.5.4 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.

Table continues on the next page...

Term	Meaning
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

Chapter 2
Introduction

2.1 Introduction

The S32R274 is a 32-bit Power Architecture based Microcontroller (MCU) unit for automotive applications. It extends the MPC5775K family by a value device optimized for surround RADAR sensors and midrange front RADAR sensors. The family members are designed to address advanced RADAR signal processing capabilities and merge it with microcontroller capabilities for generic software tasks and car bus interfacing.

S32R274 meets the high performance computation demands required by modern beam-forming fast chirp modulation RADAR systems by offering unique signal processing acceleration together with powerful multi-core architecture. The S32R274 supports automotive safety applications that require a high Safety Integrity Level (ASIL) and adds SHE compliant security features to prevent unauthorized manipulations. The high integration level of S32R274 enables the customer to built compact, safe and secure, low cost RADAR sensors with leading edge performance. The family members have a high level of software compatible for shared hardware modules.

2.1.1 Target applications

S32R274 is a high end automotive MCU designed to support computation intensive application, like ADAS RADAR.

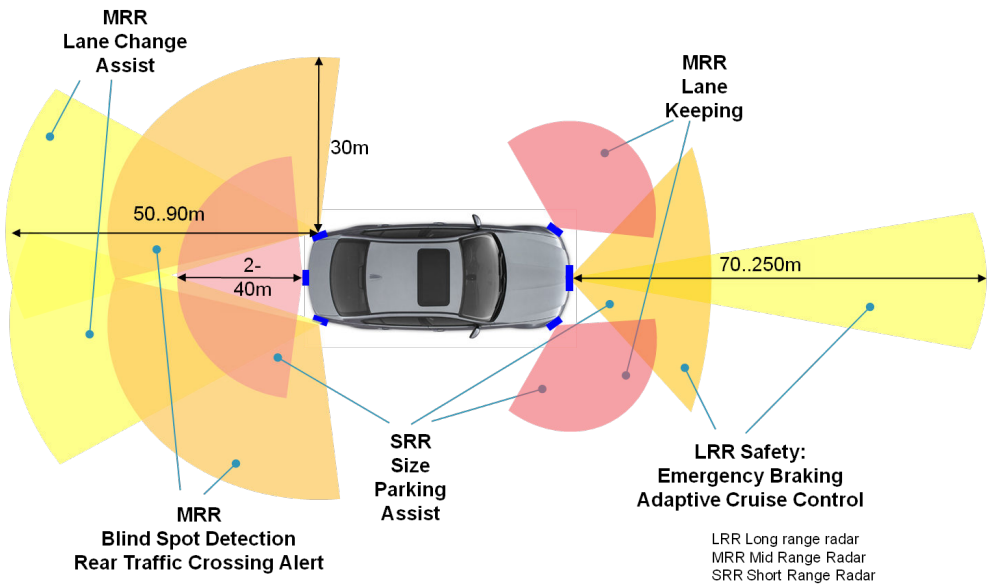


Figure 2-1. RADAR application overview

- Low to mid range RADAR
  - side-looking and surround sensors
  - lane change assist (LCA), blind-spot detection (BSD), rear-traffic-crossing alert (RTCA)
  - sampling rate <= 10MSps, 4 receive antennas
  - fast chirp modulation (FCM)
  - including clustering and tracking
- Long range value RADAR
  - forward looking sensor
  - adaptive cruise control (ACC), emergency breaking assist (EBA), pedestrian protection
  - sampling rate <= 20MSps, 4 receive antennas
  - advanced beamforming and synthetic aperture
  - fast chirp modulation (FCM)
  - including clustering and tracking and basic RADAR FUSION
- Safety application
  - using one lockstep core (z4)
  - optionally using two cores for non-safety functions (2x z7)
  - using various communication interface including Ethernet, FlexRay, FlexCAN, SPI, LINFlexD, I2C
  - optionally keeping one or both z7 cores clocked off

- Generic data processing using two cores in decoupled mode (2x z7)
  - booting from one of the z7 cores
  - keeping the safety core (z4 lockstep system) clocked off
  - optionally keeping the second z7 core clocked off
- Signal processing using SPT and Z7 SIMD units
  - Accelerated hardware FFT, vector math, peak search, statistics
  - Generic DSP functionality with SPE and VFPU unit of Z7 cores
- Security application
  - CSE hardware security engine
  - Boot and software image verfication: secure boot
  - Encryption/ decryption/ signing/ verification of messages sent or received via various communication interfaces including ENET, FlexRay, FlexCAN, SPI, LINFlexD, I2C

2.1.2 Feature list

On-chip modules available within the device include the following features:

- Safety core: Power Architecture® e200Z4 32-bit CPU with checker core
  - 2 cycle delayed lockstep
  - Harvard architecture with 64-bit bus for data and instructions
  - Dual issue: up to two instructions per clock cycle
  - 8 KB instruction cache and 4 KB data cache
  - 64 KB data local memory
    - with background load/store: backdoor access
    - 0-wait state for all read and 32/64-bit write accesses
    - Low number of wait states for backdoor accesses
  - Support for decorated storage
  - Variable Length Encoding (VLE) compliant for higher code density
  - Single precision floating point operations
- Computation cores: Power Architecture® e200Z7 32-bit CPU
  - Dual issue: up to two instructions per clock cycle
  - Harvard architecture with 64-bit bus for data instructions
  - 16 KB instruction cache and 16 KB data cache
  - 64 KB data local memory
    - with background load/store: backdoor access
    - 0-wait state for all read and 32/64-bit write accesses
    - Low number of wait states for backdoor accesses
  - Support for decorated storage
  - Using variable length encoding (VLE) for higher code density

    - 4-way integer processing unit (SPE2)
    - 2-way single-precision Floating Point Unit (EFPU2)
- 2 MB on-chip code flash (FMC flash) with ECC
    - Three ports (one per CPU) shared between code and data flash with 4 × 256 bit buffer for code and data flash including prefetch functions
    - Data flash is part of the code flash module
    - Including 64 KB EEPROM emulation
- 1.5 MB on-chip SRAM with ECC
    - Decorated memory controller to support atomic read-modify-write operations
    - Single- and double-bit error visibility is supported
    - Up to four ports (one per CPU and SPT) and up to 8 banks allow simultaneous accesses from different masters to different banks
- RADAR processing
    - Signal Processing Toolbox (SPT) for RADAR signal processing acceleration
    - Cross Timing Engine (CTE) for precise timing generation and triggering
    - Waveform generation module (WGM) for chirp ramp generation
    - 4x 12-bit ΣΔ-ADC with 10 MSps
    - One DAC with 10 MSps
    - MIPICSI2 interface to connect external ADCs
        - Four data lanes, with up to 1 Gbps per lane and in total
        - One clock lane
- Memory Protection
    - Each core memory protection unit provides 24 entries
    - Data and instruction bus system memory protection Unit (SMPU) with 16 region descriptors each
    - Register protection
- Clock Generation
    - 40 MHz external crystal (XOSC)
    - 16 MHz Internal oscillator (IRCOSC)
    - Dual system PLL with one frequency modulated phase-locked loop (FMPLL)
    - Low-jitter PLL to ΣΔ-ADC and DAC clock generation
- Functional Safety
    - Enables up to ASIL-D applications
    - End to end ECC ensuring full protection of all data accesses throughout the system, from each of the systems masters through the crossbar and into the memories and peripherals
    - FCCU for fault collection and fault handling
    - MEMU for memory error management
    - Safe eDMA controller
    - User selectable Memory BIST (MBIST) can be enabled to run out of various reset conditions or during runtime

    - Self-Test Control Unit (STCU2)
    - Error Injection Module (EIM)
    - On-chip voltage monitoring
    - Clock Monitor Unit (CMU) to support monitoring of critical clocks
- Security
    - Cryptographic Security Engine (CSE2) enabling advanced security management
    - Supports censorship and life-cycle management via Password and Device Security (PASS) module
    - Diary control for tamper detection (TDM)
- Support Modules
    - Global Interrupt controller (INTC) capable of routing interrupts to any CPU
    - Semaphore unit to manage access to shared resources
    - Two CRC computation units with four polynomials
    - 32-channel eDMA controller with multiple transfer request sources using DMAMUX
    - Boot Assist Module (BAM) supports internal flash programming via a serial link (LIN / CAN)
- Timers
    - Two Periodic Interval Timers (PIT) with 32-bit counter resolution
    - Three System Timer Module (STM)
    - Three Software Watchdog Timers (SWT)
    - Two eTimer modules with 6 channels each
    - One FlexPWM module for 12 PWM signals
- Communication Interfaces
    - Two Serial Peripheral interface (SPI) module
    - Two inter-IC communication interface (I2C) modules
    - One LINFlexD module
    - One dual-channel FlexRay module with 128 message buffers
    - Three FlexCAN modules with configurable buffers
        - CAN FD optionally supported on 2 FlexCAN modules
    - One ENET MAC supporting MII/RMII/RGMII interface
        - Supports 10/100 Mbps (MII/RMII/RGMII) and >100 Mbps (RGMII)
        - Supports IEEE1588 timestamps and PTP
    - Zipwire high-speed serial communication
        - Supports LFAST and SIPI protocol
        - Fast interprocessor communication with 320 Mbps gross data rate
        - DMA based access to memory resources
- Debug Functionality
    - 4-pin JTAG interface and Nexus/Aurora interface for serial high-speed tracing
    - e200Z7 core and e200Z4 core: Nexus development interface (NDI) per IEEE-ISTO 5001-2012 Class 3+

    - All platform bus masters except CSE can be monitored via Nexus/Aurora
    - Device/board boundary Scan testing supported with per Joint Test Action Group (JTAG) (IEEE 1149.1) and 1149.7 (cJTAG)
    - On-chip control for Nexus development interface by JTAGM module
- Two analog-to-digital converters (SAR ADC)
    - Each ADC supports up to 16 input channels
    - Cross Trigger Unit to enable synchronization of ADC conversions with eTimer
- On-chip voltage DC/DC regulator for core clock (VREG)
- Two Temperature Sensors (TSENS)

2.2 Family comparison

The following table provides a comparison of two devices S32R274 and MPC5775K . This information is intended to provide an understanding of the range of functionality offered by this family. For full details of all of the family derivatives please contact your marketing representative.

Table 2-1. S32R274 Family Comparison

Feature	S32R274	MPC5775K
CPUs	e200z420 lock-step 2x e200z7260	
SIMD	SPE2 + EFP2 (z7)	
Maximum Operating Frequency	240 MHz (z7 cores) / 180 MHz (z4)	266 MHz (z7 cores) / 133 MHz (z4)
Flash	2 MB with ECC	4 MB with ECC
EEPROM support	64 KB (emulation)	96 KB (emulation)
RAM	1.5 MB with ECC	
ECC	end-to-end	
MPU	Core MPU: 24 entries per core, System MPU: 2x16 entries	
eDMA	safe eDMA with 32 channels, 64 triggers	
Control ADC	2x 12-bit SAR ADC, 1 MSps input mux for 16 external channels	4x 12-bit SAR ADC, 1 MSps, input mux for 37 external channels
SD-ADC	4 channels, 10 MSps	8 channels, 10 MSps
SPT	1x	
CTE	1x	
WGM	1x	
CTU	1x	2x
SWT	3x	
STM	3x	
PIT	2x	
CRC	2x	

Table continues on the next page...

Table 2-1. S32R274 Family Comparison (continued)

Feature	S32R274	MPC5775K
SEMA42	1x	
LINFlexD	1x	4x
CAN	3x FlexCAN including 2x FlexCAN-FD	4x FlexCAN + 1x MCAN-FD
SPI	2x	4x
I^2C	2x	3x
Zipwire	1x LFAST+SIPI, 320 MHz	
FlexRay	1x dual channel	
Ethernet	10/100 and >100 Mbps, RMII/MII/RGMII I/F, AVB support	10/100 Mbps, RMII/MII I/F, AVB support
FlexPWM	1x, 12 PWM channels	2x, 12 PWM channels each
eTimer	2x, 6 channels each	3x, 6 channels each
External ADC interface	1x 4 lanes MIPICSI2 Rx, 1 Gbps/lane	1x PDI (16-bit data, clock, sync)
IRCOSC	16 MHz	
XOSC	40 MHz	
FMPLL	dual system PLL, 1x FM modulated	
DAC	1x 12-bit 10 MSps	1x 12-bit 2 MSps
SIUL2	1x	
BAM	1x	
INTC	1x	
SSCM	1x	
FCCU/FOSU	1x	
MEMU	1x	
STCU2	1x	
CSE	1x	-
PASS/TDM	1x	-
MC_ME	1x	
MC_CGM	1x	
MC_RGM	1x	
TSENS	2x	
Debug	JTAGC, JTAGM, CJTAG, with class3+ Nexus, Aurora only	
Safety level	ISO26262 SEooC ASIL-B to ASIL-D	
Temp. range (Tj)	-40 to 150˚C	

2.3 Block diagram

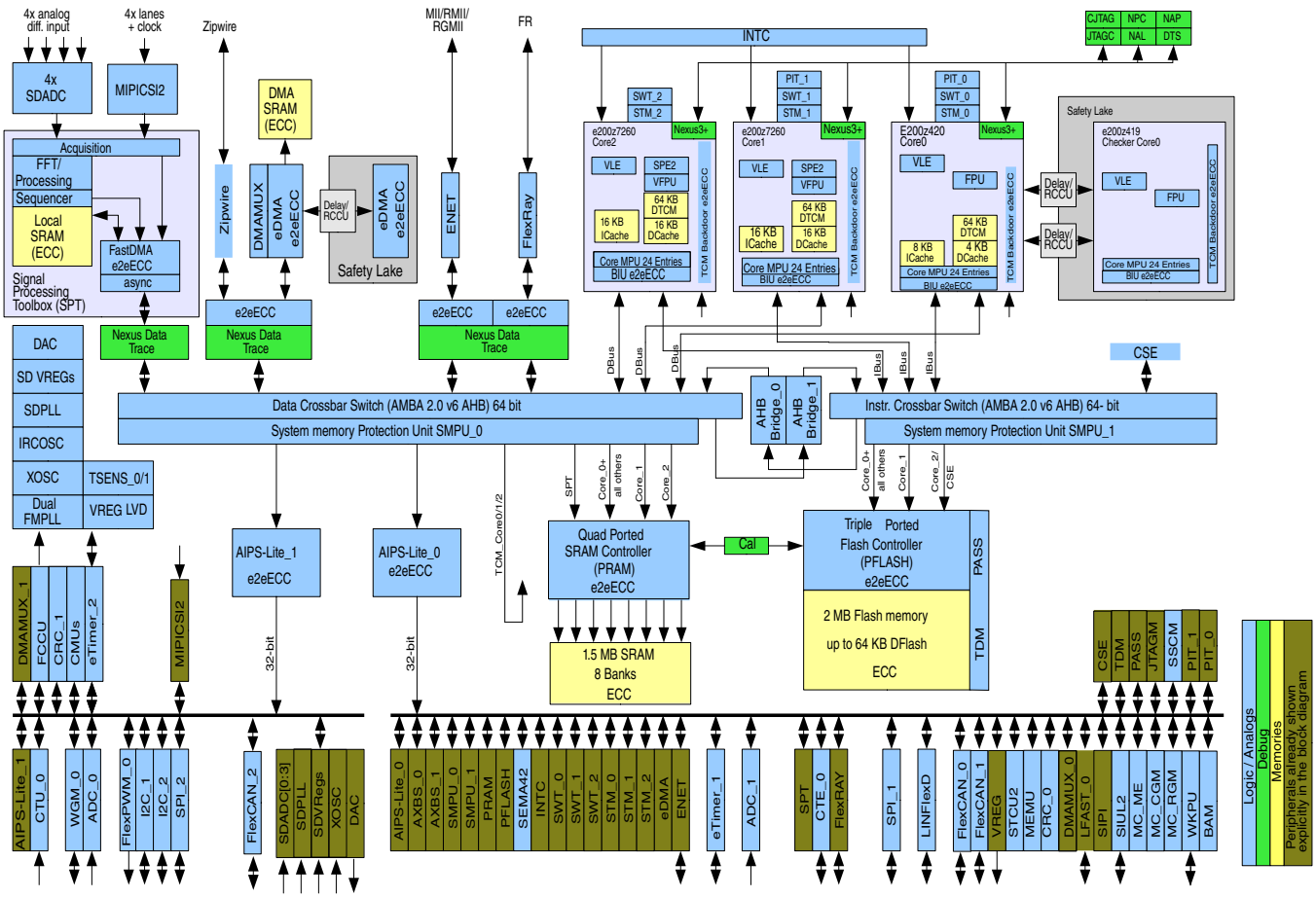


Figure 2-2. S32R274 block diagram

NOTE

DTCM corresponds to DMEM. For details, see Core chapters in this document.

2.4 Package

The chip family member is offered in the following package type:

- 257-ball MAPBGA, 0.8 mm ball pitch, 14 mm x 14 mm outline

Chapter 3
Memory Map

3.1 Introduction

See the device Memory map tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click on it.

Chapter 4
Signal Multiplexing and Signal Description

4.1 Generic pins/balls

The I/O Signal Description Table contains information on pins/balls of this device. See the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

4.2 Pad states

The following table shows the pad types and states.

Table 4-1. Pad states

PADs	Type	POR[1]	RESET	SELF TEST	RUN
PAD_0	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_1	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_2	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_3	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_4	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_5	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_6	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_7	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_8	GPIO	high impedance	weak pull down	high impedance	weak pull down
PAD_9	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_10	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_11	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_12	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_13	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_14	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_15	GPIO	high impedance	weak pull down	high impedance	weak pull down

Table continues on the next page...

Table 4-1. Pad states (continued)

PADs	Type	POR[1]	RESET	SELF TEST	RUN
PAD_16	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_17	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_18	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_19[2]	GPIO	high impedance	high impedance	weak pull-up	high impedance
PAD_20	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_21	GPIO	high impedance	weak pull-up	weak pull-up	weak pull-up
PAD_22	GPIO	high impedance	high impedance	high impedance	high impedance
ADC0_AN_0	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_1	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_11_ADC1_AN_11	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_12_ADC1_AN_12	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_13_ADC1_AN_13	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_14_ADC1_AN_14	Analog	high impedance	high impedance	high impedance	high impedance
ADC1_AN_0	Analog	high impedance	high impedance	high impedance	high impedance
ADC1_AN_1	Analog	high impedance	high impedance	high impedance	high impedance
ADC1_AN_2	Analog	high impedance	high impedance	high impedance	high impedance
ADC1_AN_3	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_2	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_3	Analog	high impedance	high impedance	high impedance	high impedance
PAD_42	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_43	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_44	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_47	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_48	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_49	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_50	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_51	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_52	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_53	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_54	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_56	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_59	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_62	GPIO	high impedance	high impedance	high impedance	high impedance
ADC0_AN_5	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_7	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_8	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_4	Analog	high impedance	high impedance	high impedance	high impedance
ADC0_AN_6	Analog	high impedance	high impedance	high impedance	high impedance
PAD_77	GPIO	high impedance	high impedance	high impedance	high impedance

Table continues on the next page...

Table 4-1. Pad states (continued)

PADs	Type	POR[1]	RESET	SELF TEST	RUN
PAD_79	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_80	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_94	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_95	GPIO	high impedance	high impedance	high impedance	high impedance
FCCU_F_0	GPIO	high impedance	high impedance	high impedance	high impedance
FCCU_F_1	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_101	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_104	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_105	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_106	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_107	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_116	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_117	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_118	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_119	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_120	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_121	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_123	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_124	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_125	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_128	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_129	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_130	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_132	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_133	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_134	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_135	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_136	GPIO	high impedance	high impedance	high impedance	high impedance
PAD_137	GPIO	high impedance	high impedance	high impedance	high impedance
NMI	GPIO	high impedance	weak pull-up	weak pull-up	weak pull-up
RESET_B	GPIO	DRIVE LOW	DRIVE LOW	undefined[3]	weak pull-down
TCK	GPIO	high impedance	weak pull-up	weak pull-up	weak pull-up
TMS	GPIO	high impedance	weak pull-up	weak pull-up	weak pull-up
JCOMP	GPIO	high impedance	weak pull-down	weak pull-down	weak pull-down

1. Pads marked as high impedance will be in either high-impedance or weak low drive state when VDD_LV_CORE is off and HV_VDD_IO is below 1.5V.
2. When VDD_HV_PMC/VDD_HV_IO supply is powered, while any other supply holds the part in reset, as well as VREG_POR_B is HIGH, then PAD_19 can become pull up [i.e. the state of pad during self-test] briefly before going to Hi-Z [i.e. the state of the pad in Run/Reset].
3. Could be drive low if running offline STCU or pull-down if running online STCU

NOTE

Define 'Dext_pull_level' as the device PULL UP on a pad in the RUN column. 'Dext_pull_level' is '1' for weak PULL UP, '0' for weak pull down, and 'Hi-Z' for high-impedance.

If the user's implementation requires an external pull up or pull down (Pext_pull_level) on each pad to SAFE state then the following options could be executed:

- First priority is to implement an external pull level ('1' or '0') as per the protocol planned to be used in end application. For example: PAD_119 is used as I2C_1 SCLK then the preferred level of external pull is high as per the usage (Pext_pull_level='1'). See IO Signal Description and Input multiplexing tables attached as an excel file with this document.
- Second priority is to implement an external pull level ('1' or '0') (Pext_pull_level) to same level as the default design pull level (Dext_pull_level) on that pad (Pext_pull_level=Dext_pull_level). Following this guidelines would save on minor leakage on a pad in case if an external pull level (Pext_pull) is opposite to the pull level in RUN mode (Dext_pull).

4.3 GPIO safe state configuration

The following table shows the GPIOs behavior when chip is in SAFE mode. It is dependent on the SIUL2_MSCR[SMC], FCCU in fault state and MC_ME_SAFE_MC[PDO].

SIUL2_MSCR[SMC]	FCCU Fault State	MC_ME_SAFE_MC[PDO]	GPIO (SIUL)	Dedicated FCCU EOUT pads[1]
1	0	0	Not Hiz	Not Hiz
1	0	1	Not Hiz	Not Hiz
1	1	0	Not Hiz	Not Hiz
1	1	1	Not Hiz	Not Hiz
0	0	0	Not Hiz	Not Hiz
0	0	1	Hiz	Not Hiz
0	1	0	Hiz	Not Hiz
0	1	1	Hiz	Not Hiz

1. Fault output (EOUT) signalling should be enabled. See FCCU

4.4 Special cases of IBE and OBE control

The following table lists the input buffer enable[IBE] and output buffer enable[OBE] configurations of all pads in the device.

Table 4-2. Special cases of IBE and OBE control

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[0]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	SCK	O	module	-	-
WGM	WG0	O	Always ON	-	-
SIUL2	EIRQ[0]	I	-	SIUL2_MSCRn	-
SPI_2	SCK	I	-	SIUL2_MSCRn	
SIUL2	GPIO[1]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	SOUT	O	module	-	-
WGM	WG1	O	Always ON	-	-
SIUL2	EIRQ[1]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[2]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	A3	O	module	-	-
WGM	WG2_P	O	Always ON	-	-
SIUL2	EIRQ[2]	I	-	SIUL2_MSCRn	-
SPI_2	SIN	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[3]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS0	O	module	-	-
FlexPWM_0	B3	O	module	-	-
WGM	WG2_N	O	Always ON	-	-
SIUL2	EIRQ[3]	I	-	SIUL2_MSCRn	-
SPI_2	CS0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[4]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_1	ETC0	O	module	-	-
SPI_2	CS1	O	module	-	-
SSCM	DEBUG4	O	Always ON	-	-
SIUL2	EIRQ[4]	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[5]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	CS0	O	module	-	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
eTIMER_1	ETC5	O	module	-	-
SSCM	DEBUG5	O	Always ON	-	-
SIUL2	EIRQ[5]	I	-	SIUL2_MSCRn	-
SPI_1	CS0	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC5	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[6]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	SCK	O	module	-	-
eTIMER_2	ETC2	O	module	-	-
CTE	CTEP0	O	module	-	-
SSCM	DEBUG6	O	Always ON		-
SIUL2	EIRQ[6]	I	-	SIUL2_MSCRn	-
SPI_1	SCK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC2	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[7]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	SOUT	O	module	-	-
eTIMER_2	ETC3	O	module	-	-
CTE	CTEP1	O	module	-	-
SSCM	DEBUG7	O	Always ON	-	-
SIUL2	EIRQ[7]	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC3	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[8]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE ON,OBE OFF
eTIMER_2	ETC4	O	module	-	-
CTE	CTEP2	O	module	-	-
SSCM	FAB	I	-	SIUL2_MSCRn	-
SIUL2	EIRQ[8]	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC4	I	-	SIUL2_MSCRn	-
SPI_1	SIN	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[9]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS1	O	module	-	-
eTIMER_2	ETC5	O	module	-	-
FlexPWM_0	B3	O	module	-	-
SPI_1	SOUT	O	module	-	-
eTIMER_2	ETC5	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[0]	I	-	SIUL2_MSCRn	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[16]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexCAN_0	TXD	O	Always ON	-	-
eTIMER_1	ETC2	O	module	-	-
SSCM	DEBUG0	O	Always ON	-	-
SIUL2	EIRQ[15]	I		SIUL2_MSCRn	-
eTIMER_1	ETC2	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[17]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexCAN_0	TXD	O	Always ON	-	-
eTIMER_1	ETC3	O	module	-	-
SSCM	DEBUG1	O	Always ON	-	-
SIUL2	EIRQ[16]	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC3	I	-	SIUL2_MSCRn	-
FlexCAN_0	RXD	I	-	SIUL2_MSCRn	-
FlexCAN_1	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[18]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SSCM	DEBUG2	O	Always ON	-	-
SPI_1	SCK	O	module	-	-
SIUL2	EIRQ[17]	I	-	SIUL2_MSCRn	-
SPI_1	SCK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[19]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SSCM	DEBUG3	O	Always ON	-	-
SPI_1	SIN	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[20]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF,OBE ON
JTAGC	TDO	O	module	-	
SIUL2	GPIO[21]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE ON,OBE OFF
JTAGC	TDI	I		SIUL2_MSCRn	
SIUL2	GPIO[22]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
MC_CGM	CLK_OUT0	O	Always ON	-	-
SPI_2	CS2	O	module	-	-
SIUL2	EIRQ[18]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[23]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[24]	I	-	SIUL2_MSCRn	Default IBE Off

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[25]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[32]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	EIRQ[22]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[33]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	EIRQ[23]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[34]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	EIRQ[24]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[48]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	TXD_A	O	Always ON	-	-
eTIMER_1	ETC1	O	module	-	-
FlexPWM_0	B1	O	module	-	-
ENET	TX_D0	O	Always ON	-	-
eTIMER_1	ETC1	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[49]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_1	ETC2	O	module	-	-
CTU_0	EXT_TRG	O	Always ON	-	-
FLEXRAY	CA_RX	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC2	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC1	I	-	SIUL2_MSCRn	-
ENET	RX_CLK	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[50]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_1	ETC3	O	module	-	-
FlexPWM_0	X3	O	module	-	-
FLEXRAY	CB_RX	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC3	I	-	SIUL2_MSCRn	-
FlexPWM_0	X3	I	-	SIUL2_MSCRn	-
ENET	RX_DV	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[51]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	TXD_B	O	Always ON	-	-
eTIMER_1	ETC4	O	module	-	-
FlexPWM_0	A3	O	module	-	-
ENET	TX_D1	O	Always ON	-	-
eTIMER_1	ETC4	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[52]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
FLEXRAY	TXEN_B	O	Always ON	-	-
eTIMER_1	ETC5	O	module	-	-
FlexPWM_0	B3	O	module	-	-
ENET	TX_D2	O	Always ON	-	-
eTIMER_1	ETC5	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[53]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
ENET	RX_D0	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[2]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[54]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	X3	O	module	-	-
ENET	RX_D1	I	-	SIUL2_MSCRn	-
FlexPWM_0	X3	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[1]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[56]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	CS2	O	module	-	-
eTIMER_1	ETC4	O	module	-	-
FlexPWM_0	X0	O	module	-	-
WGM	WG3_P	O	Always ON	-	-
eTIMER_1	ETC4	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[3]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[66]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[68]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[69]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[70]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[71]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[80]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	A1	O	module	-	-
ENET	TIMER2	O	module	-	-
CTE	RCS	O	module	-	-
SIUL2	EIRQ[28]	I	-	SIUL2_MSCRn	-
CTE	RCS	I	-	module and SIUL2_MSCRn	-
SIUL2	GPIO[101]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	X3	O	module	-	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SPI_2	CS3	O	module	-	-
CTE	OVFL	I	-	SIUL2_MSCRn	-
FlexPWM_0	X3	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[104]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	DEBUG_0	O	Always ON	-	-
ENET	MDIO	O	module	-	-
SIUL2	EIRQ[21]	I	-	SIUL2_MSCRn	-
ENET	MDIO	I	-	SIUL2_MSCRn	
FlexPWM_0	FAULT[0]	I	-	SIUL2_MSCRn	
SIUL2	GPIO[105]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	DEBUG_1	O	Always ON	-	-
SPI_1	CS1	O	module	-	-
ENET	MDC	O	Always ON	-	-
SIUL2	EIRQ[29]	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[1]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[116]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC0	O	module	-	-
ENET	RX_D2	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[117]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
ENET	RX_D3	I	-	SIUL2_MSCRn	
SIUL2	GPIO[118]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	CS0	O	module	-	-
CTE	CTEP5	O	module	-	-
ENET	COL	I	-	SIUL2_MSCRn	-
SPI_1	CS0	I	-	SIUL2_MSCRn	
SIUL2	GPIO[119]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC1	O	module	-	-
I2C_1	CLK	O	module	-	-
ENET	CRS	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC1	I	-	SIUL2_MSCRn	-
I2C_1	CLK	I	-	SIUL2_MSCRn	-
SPI_1	SIN	I	-	SIUL2_MSCRn	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[120]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
I2C_1	DATA	O	module	-	-
ENET	RX_ER	I	-	SIUL2_MSCRn	-
I2C_1	DATA	I		SIUL2_MSCRn	-
SIUL2	GPIO[121]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
ENET	TX_ER	O	Always ON	-	-
SIUL2	GPIO[128]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC0	O	module	-	-
CTE	CTEP5	O	module	-	-
SPI_1	CS7	O	module	-	-
eTIMER_2	ETC0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[129]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC1	O	module	-	-
SPI_1	CS5	O	module	-	-
ENET	TIMER0	O	module	-	-
SIUL2	EIRQ[26]	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC1	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[130]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC2	O	module	-	-
ENET	REF_CLK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC2	I	-	SIUL2_MSCRn	-
ENET	RX_DV	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[132]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
NPC	NEX_RDY_B	O	Always ON	-	-
SIUL2	GPIO[133]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexCAN_2	TXD	O	Always ON	-	-
I2C_2	CLK	O	module	-	-
SPI_1	CS2	O	module	-	-
FLEXRAY	TXEN_A	O	Always ON	-	-
I2C_2	CLK	I	-	SIUL2_MSCRn	-
FlexCAN_2	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[134]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
FlexCAN_1	TXD	O	Always ON	-	-
I2C_2	DATA	O	module	-	-
SPI_1	CS6	O	module	-	-
FLEXRAY	TXD_A	O	Always ON	-	-
I2C_2	DATA	I	-	SIUL2_MSCRn	-
FlexCAN_2	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[135]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
MC_CGM	LFAST_REF_CLK	O	module	-	-
eTIMER_2	ETC2	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[136]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_1	CS7	O	module		-
NPC	EVTI_IN	I	-	SIUL2_MSCRn	-
CTE	RCS	I	-	module and SIUL2_MSCRn	-
SIUL2	GPIO[137]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC4	O	module	-	-
SPI_1	CS2	O	module	-	-
NPC	EVTO_B	O	module	-	-
eTIMER_2	ETC4	I	-	SIUL2_MSCRn	-
CTE	RCS	I	-	module and SIUL2_MSCRn	-
SIUL2	GPIO[10]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS0	O	module	-	-
FlexPWM_0	B0	O	module	-	-
FlexPWM_0	X2	O	module	-	-
SIUL2	EIRQ[9]	I	-	SIUL2_MSCRn	-
SPI_2	CS0	I	-	SIUL2_MSCRn	-
FlexPWM_0	X2	I		SIUL2_MSCRn	-
SIUL2	GPIO[11]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	SCK	O	module	-	-
FlexPWM_0	A0	O	module	-	-
FlexPWM_0	A2	O	module	-	-
SIUL2	EIRQ[10]	I	-	SIUL2_MSCRn	-
SPI_2	SCK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC4	I	-	SIUL2_MSCRn	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[12]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	SOUT	O	module	-	-
FlexPWM_0	A2	O	module	-	-
FlexPWM_0	B2	O	module	-	-
SIUL2	EIRQ[11]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[13]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	B2	O	module	-	-
SIUL2	EIRQ[12]	I	-	SIUL2_MSCRn	-
SPI_2	SIN	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[0]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[14]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexCAN_1	TXD	O	Always ON	-	-
eTIMER_1	ETC4	O	module	-	-
SIUL2	EIRQ[13]	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC4	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[15]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE ON,OBE OFF
eTIMER_1	ETC5	O	module	-	-
SSCM	ABS[0]	I	-	SIUL2_MSCRn	-
SIUL2	EIRQ[14]	I	-	SIUL2_MSCRn	-
eTIMER_1	ETC5	I	-	SIUL2_MSCRn	-
FlexCAN_0	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[26]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[27]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[28]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	GPIO[29]	I	-	SIUL2_MSCRn	Default IBE Off
LINFLexD_1	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[30]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	EIRQ[19]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[31]	I	-	SIUL2_MSCRn	Default IBE Off
SIUL2	EIRQ[20]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[42]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS2	O	module	-	-
FlexPWM_0	A3	O	module	-	-
CTE	CTEP3	O	module	-	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
FlexPWM_0	FAULT[1]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[43]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS2	O	module	-	-
CTE	CTEP4	O	module	-	-
SIUL2	GPIO[44]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
SPI_2	CS3	O	module	-	-
eTIMER_2	ETC3	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[47]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	TXEN_A	O	Always ON	-	-
eTIMER_1	ETC0	O	module	-	-
FlexPWM_0	A1	O	module	-	-
ENET	TX_EN	O	Always ON	-	-
eTIMER_1	ETC0	I	-	SIUL2_MSCRn	-
CTU_0	EXT_IN	I	-	SIUL2_MSCRn	-
FlexPWM_0	EXT_SYNC	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[59]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	B0	O	module	-	-
LINFLexD_1	TXD	O	Always ON	-	-
FlexPWM_0	X1	O	module	-	-
WGM	WG3_N	O	Always ON	-	-
FlexPWM_0	X1	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[62]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	B1	O	module	-	-
SPI_1	CS3	O	module	-	-
CTU_0	EXT_TRG	O	Always ON	-	-
SPI_1	CS7	O	module	-	-
SIUL2	GPIO[77]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FlexPWM_0	X1	O	module	-	-
SPI_2	CS3	O	module	-	-
SPI_1	CS4	O	module	-	-
CTE	RFS	O	module	-	-
SIUL2	EIRQ[25]	I	-	SIUL2_MSCRn	-
CTE	RFS	I	-	module and SIUL2_MSCRn	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
FlexPWM_0	X1	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[79]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
MC_CGM	CLK_OUT1	O	Always ON	-	-
ENET	TIMER1	O	module	-	-
SPI_1	CS0	O	module	-	-
ENET	TIMER1	I	-	SIUL2_MSCRn	-
SIUL2	EIRQ[27]	I	-	SIUL2_MSCRn	-
SPI_1	CS0	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[94]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
LINFLexD_1	TXD	O	Always ON	-	-
FlexCAN_2	TXD	O	Always ON	-	-
I2C_1	CLK	O	module	-	-
ENET	TS1588_CLK	I	-	SIUL2_MSCRn	-
I2C_1	CLK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC4	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[95]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
I2C_1	DATA	O	module	-	-
I2C_1	DATA	I	-	SIUL2_MSCRn	-
LINFLexD_1	RXD	I	-	SIUL2_MSCRn	-
FlexCAN_2	RXD	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[106]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	DEBUG_2	O	Always ON	-	-
ENET	TX_D3	O	Always ON	-	-
SIUL2	EIRQ[30]	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[2]	I	-	SIUL2_MSCRn	-
SIUL2	GPIO[107]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
FLEXRAY	DEBUG_3	O	Always ON	-	-
ENET	TXC	O	Always ON	-	-
ENET	RMII_CLK	O	module (MCB)	-	-
SIUL2	EIRQ[31]	I	-	SIUL2_MSCRn	-
ENET	RMII_CLK	I	-	SIUL2_MSCRn	-
eTIMER_2	ETC5	I	-	SIUL2_MSCRn	-
FlexPWM_0	FAULT[3]	I	-	SIUL2_MSCRn	-
ENET	TX_CLK	I	-	SIUL2_MSCRn	-

Table continues on the next page...

Table 4-2. Special cases of IBE and OBE control (continued)

Module	Function[1]	Direction	OBE Control via	IBE Control via	Notes
SIUL2	GPIO[123]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
CTE	CTEP6	O	module	-	-
SPI_1	CS1	O	module	-	-
SIUL2	GPIO[124]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
CTE	CTEP7	O	module	-	-
SPI_1	CS5	O	module	-	-
SIUL2	GPIO[125]	I/O	SIUL2_MSCRn	SIUL2_MSCRn	Default IBE OFF/OBE OFF
eTIMER_2	ETC3	O	module	-	-
eTIMER_2	ETC3	I	-	SIUL2_MSCRn	-
ENET	RX_CLK	I	-	SIUL2_MSCRn	-
FCCU	F0	I/O	module	Always ON	-
FCCU	F1	I/O	module	Always ON	-
WKPU	INP	I	-	Always ON	-
RGM	RESET	I/O	module	Always ON	-
JTAGC	TCK	I	-	Always ON	-
JTAGC	TMS	I/O	module	Always ON	-
JTAGC	JCOMP	I	-	Always ON	-

1. For details on source signal connection to the associated destination, refer "SSS" column in 'IO signal table' tab in the I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document.

4.5 PCB routing guidelines

Following are the PCB routing guidelines applicable for this chip.

4.5.1 Decoupling Capacitors

- For detailed decoupling capacitor values and placement recommendations please see AN5251 - Hardware Design Guide.

4.5.2 MIPICSI2

- Route all signals on top and bottom layer only.
- Route all the signals of single instance in same PCB layer.

- Use low loss dielectric constant material in PCB stack up for MIPICSI2 signal adjacent layers e.g. megtron.
- Route the MIPICSI2 signals 'P' and 'N' channel as differential pair with control impedance of 100 ohm.
- The differential pairs should be matched in less than a mil.
- All the signals should be matched with tight tolerance preferably 1 mil.

4.5.3 LVDS (AURORA and LFAST)

- The LVDS pairs should be routed as differential pair with control impedance of 100 ohm and tolerance of 10 mil.
- All the signals of an interface should be tightly matched to less than 1% of the total length.

4.5.4 SAR-ADC

- Use single ground strategy for board
- The VrefP and VrefN nets should have at least one decap soldered direct on the chip pin.
- The VrefP and VrefN should be short and clean trace and not cross digital section or splits.
- The analog section should be at least 200 mil away from any digital component.
- Route ADC channels on inner layer with ground layer on both sides.
- Use thick trace or shape to provide supply to SARADC.

4.5.5 Power supply

- Use large shape to feed power for all interfaces. The thickness of shape should be at least greater than 200 mil.
- Place the smallest value decap directly on the chip pin between the VDD (supply) and VSS (ground) for each interface.

Chapter 5
Clocking Overview

5.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive Clock Switching

NOTE

This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The clock (system clock) for the computational shell (which includes the three CPUs as well as all modules that run in the high speed domain), memories, and debug logic is independent from the peripheral clocks. This allows the system clock to be a frequency modulated clock to reduce electromagnetic emissions, while having a precise clock for the peripheral timer and communication functions. It also allows the system clock to be reduced during low computational activity periods to help reduce power consumption, while providing a fixed frequency to the communication interfaces and timers.

The device boots from the internal 16 MHz RC oscillator (IRCOSC), and uses this as a backup clock in the event of a PLL or oscillator failure (if backup clock is enabled). The backup clock is enabled in the Mode Entry Module (MC_ME).

There are three possible ways to provide the source clock:

- External oscillator

- External crystal
- IRCOSC

From one of these input sources, the internal clocks are generated from one of the two PLLs, PLL0 and PLL1, PLL0_PHI_CLK and PLL1_PHI_CLK, respectively. These two clocks, along with the XOSC_CLK, IRC_CLK, and SDPLL_CLK, can be selected to drive system peripherals depending on the configuration of the Auxiliary Clock Selectors in the Clock Generation Module (MC_CGM). The PHI1 output of PLL0 (PLL0_PHI1_CLK) can also be used as the clock source for PLL1. There are a total of 15 clock selectors which allows developers to drive each of system peripherals with an independent clock source. There is also one additional clock selector which is used exclusively for the system clock.

Each of the outputs of the module clock selectors have their own divider which allows for even more clock frequency granularity being able to divide the outputs by up to 512 for a given peripheral. On selected outputs of the clock dividers are Clock Monitor Units (CMU) that are used to test the integrity of the clocks making sure that their frequencies stay within necessary operating limits. If any of the CMUs detect an issue with the clock signal that is being monitored, an interrupt or system reset could be generated, depending on how the CMUs are configured.

5.2 Clock generation

The top-level clock generation architecture is given in the following figure. All clock selectors and dividers in the figure are located and programmed in the Clock Generation Module (MC_CGM). All dividers are integer dividers (1, 2, 3, …, n) with the ranges given in the diagram. All dividers connected to the System Clock Selector in the figure have 50% duty cycle outputs.

NOTE

All references to SDPLL_CLK would refer to SDPLL_CLK160 unless explicitly mentioned otherwise.

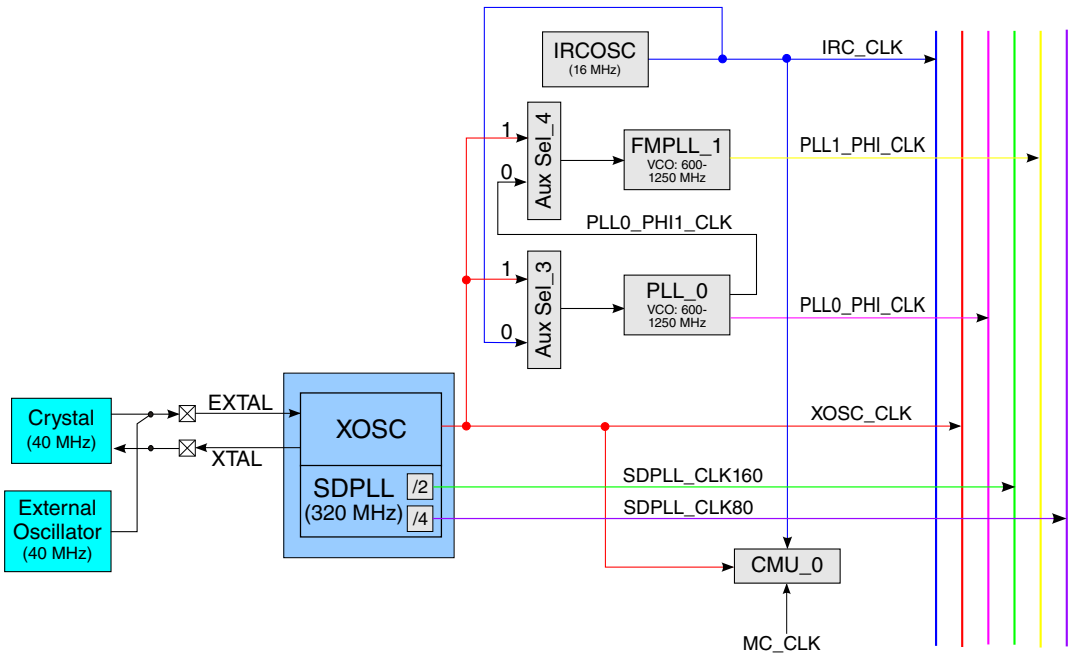


Figure 5-1. Clock sources

Figure 5-1 shows the potential clock sources. The IRC_CLK is a reliable clock source, that runs independently from any other component. It is the main clock source during boot or in case of lock loss in the PLL.

During Run mode, the clock generation is typically based on the XOSC. The XOSC module requires an external oscillator source, either a crystal or an external clock generator. For use with the Sigma Delta ADCs, this external crystal needs to be 40 MHz. If a 40 MHz crystal is used, PLL0 and SDPLL are driven by the XOSC. SDPLL_CLK is primarily used for the 320 MHz Sigma Delta ADCs and the analog RADAR system.

PLL0_PHI_CLK is used to drive the clock for the digital system. PLL1 is driven by the either the XOSC_CLK or PLL0:PHI1. PLL1 also supports Frequency Modulation to distribute the emitted electromagnetic radiation (EMR) over a frequency spectrum.

The following clocks are available from the clock source structure for individual clock domains (see Figure 5-1):

- IRC_CLK
- XOSC_CLK
- PLL0_PHI_CLK
- PLL1_PHI_CLK
- SDPLL_CLK

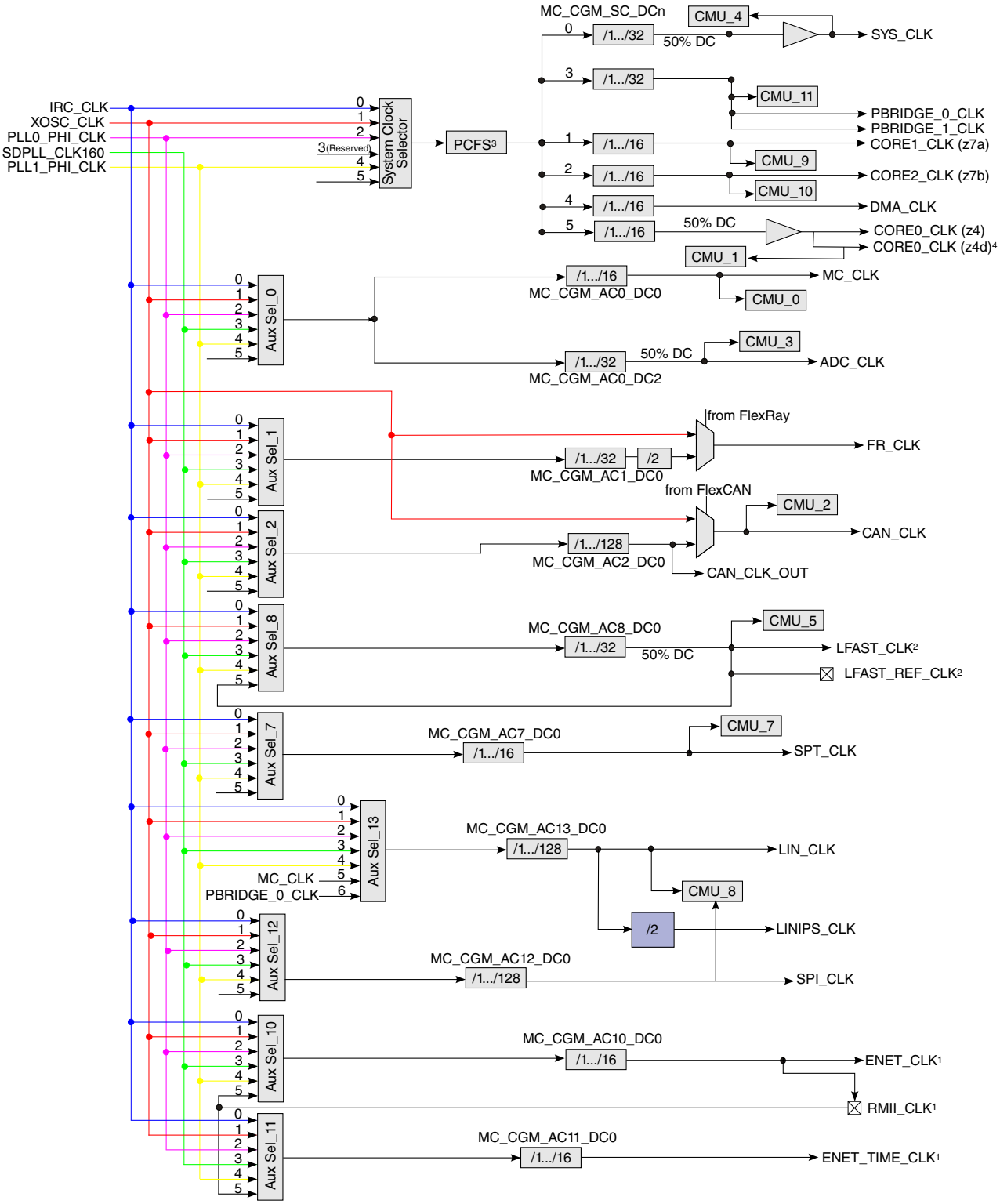


Figure 5-2. Clock selection

NOTE

1. See ENET clocking for details.
2. See LFAST clocking for details.

3. Progressive Clock Frequency Switching (see Progressive System Clock Switching for details).
4. This is the lock step core clock which is gated when lock step is disabled.

Figure 5-3 shows how the CLKOUT signals are selected using the MCB_CLKOUT_SEL register.

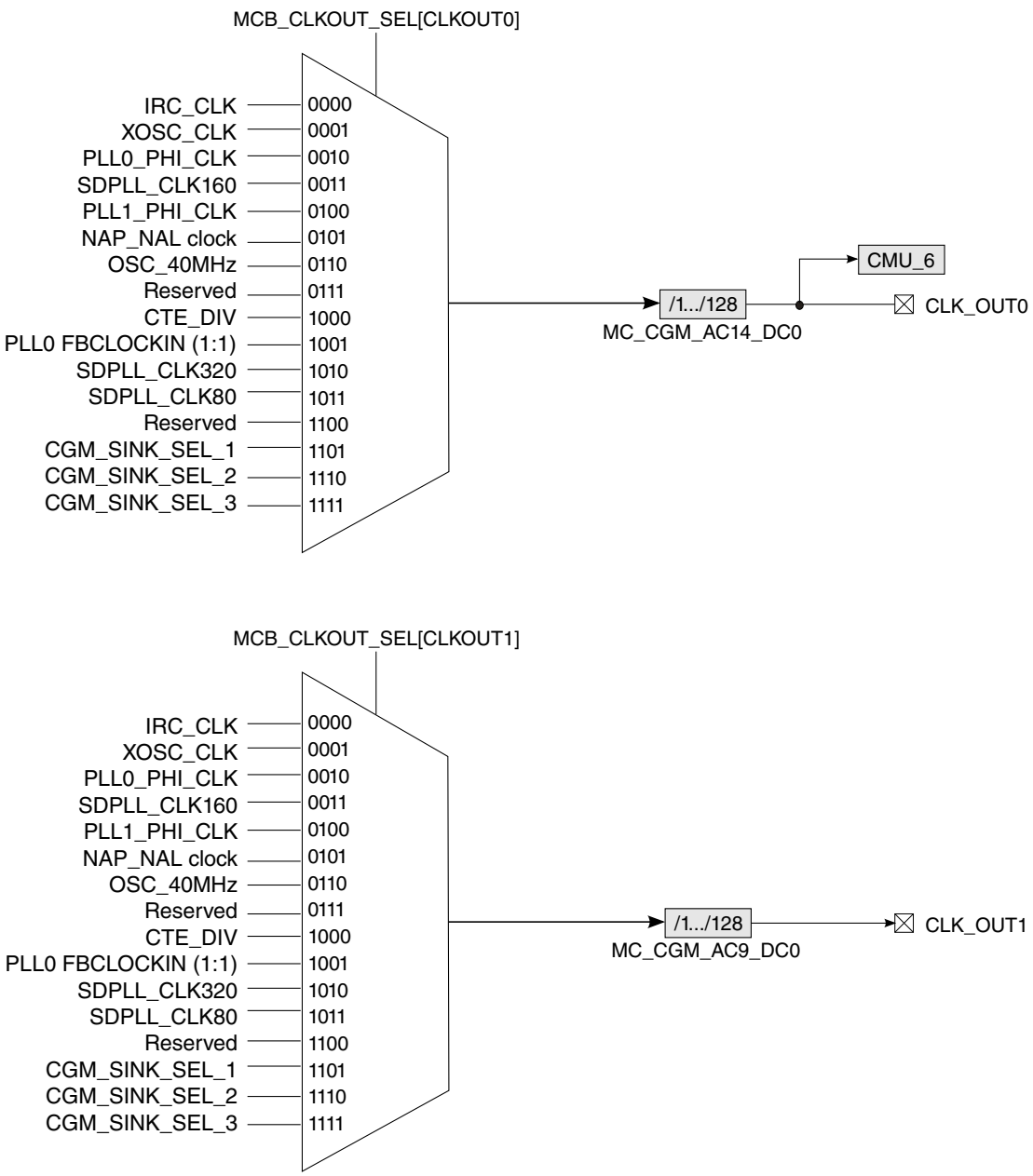


Figure 5-3. CLKOUT selection

The main clock is the system clock (SYS_CLK). It drives the Safety core and the crossbar. It is a divided version of the progressive clock switching (PCFS) clock output. Besides the System clock there are two more CPU clocks, the CORE1_CLK and CORE2_CLK for the two decoupled cores. They can be selected to be the same as the SYS_CLK or full speed clock coming from the PCFS output directly without division. For reliability reasons the CORE0_CLK has an independent clock tree and independent buffering, thus, reducing common cause faults (CCF). The CORE0_CLK is monitored by CMU_1. SYS_CLK is divided by 2, and this reduced clock is used for other XBAR masters (eDMA, LFAST, ENET, and FlexRay including the required port splitters) and for the two PBRIDGE peripheral bus bridges (PBRIDGE_0, PBRIDGE_1). The frequencies of the system clock structure, including the peripheral clocks (DMA_CLK, CORE2_CLK, CORE1_CLK, PBRIDGE_0_CLK, and PBRIDGE_1_CLK), can be gradually reduced using the fine granularity offered by the progressive clock frequency switching module (see the PCFS block in Figure 5-2). This is based on a fractional clock divider allowing smooth ramping of the current of the CPU and peripherals used for basic operation.

Besides the system clock domain, there are several other clock domains that do not allow progressive clock switching or frequency modulation. They are used for communication interfaces (FR_CLK, CAN_CLK, LFAST_CLK), for the motor control system (MC_CLK and ADC_CLK), and RADAR processing (SD_CLK, DAC_CLK, CTE_CLK, SPT_CLK). MC_CLK drives Motor Control system modules CTU_0, ETIMER_1, ETIMER_2, flexPWM_0.

CLK_OUT*n* drive the clk_out*n* pin, which can be used to monitor internal clocks (divided down so that the clk_out*n* is < 60 MHz) or to share the clock driving other chips.

Most of the clocks are clock monitored using CMU modules detecting missing or faulty clocks. The reference for the CMUs is the IRCOSC. The IRC_CLK itself is supervised by the external clock (XOSC_CLK).

Figure 5-4 shows how the clocks from the clock selectors are connected to the system components. All BIUs (bus interface units to the PBRIDGE bus) are running on PBRIDGE_0_CLK or PBRIDGE_1_CLK. These clocks are only active at the peripherals on active bus cycles to this particular module.

Thus, all modules also have a module clock that is typically a gated (per module) version of the PBRIDGE_*n*_CLK. Besides this, the software watchdog timer SWT uses the more reliable internal RC oscillator clock IRC_CLK. This clock is always on, and thus, well apt for safety functions.

The IRC_CLK may be used for safety relevant functions in the Fault Collection and Control Unit (FCCU). Furthermore, this clock is used for the digital filters to detect external interrupt events. Thus, the system can resume from STOP mode, while the system clock is gated off and only the IRC_CLK is running.

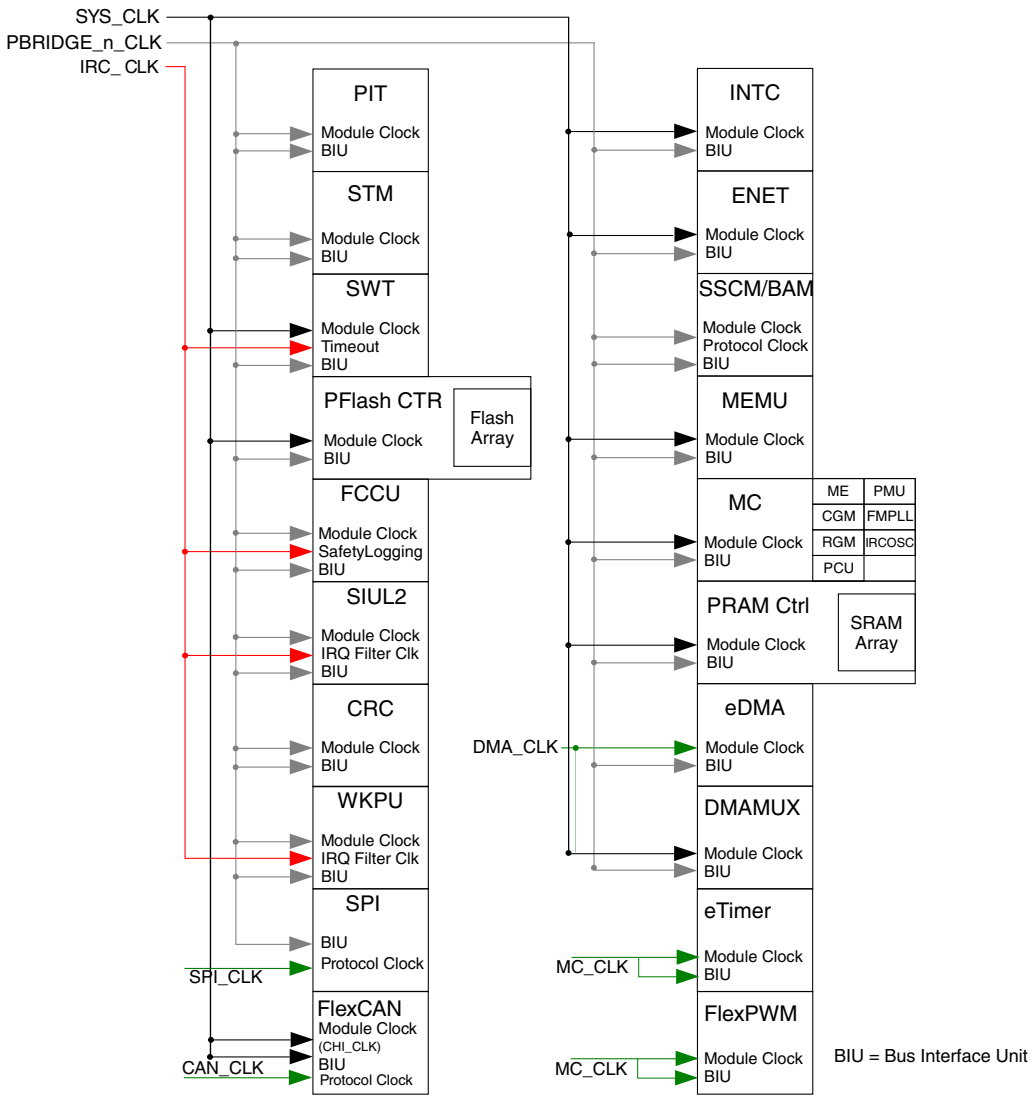


Figure 5-4. Clock distribution

The functional clocks MC_CLK, ADC_CLK, FR_CLK, CAN_CLK, LFAST_CLK, SD_CLK, DAC_CLK, CTE_CLK, and SPT_CLK are routed to the corresponding IP modules.

Special care needs to be taken to the clock distribution in the high fidelity RADAR analog subsystem (see Figure 5-5).

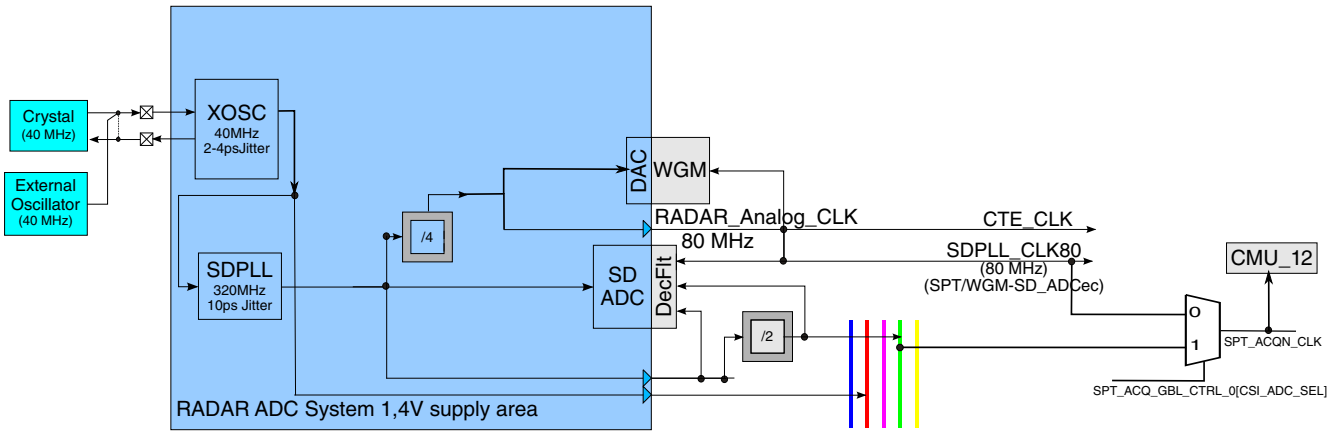


Figure 5-5. Clocks in the analog RADAR block

5.2.1 MC_CGM registers

The following table shows the relationship between the output clocks and the MC_CGM registers.

NOTE

See Clock Generation Module (MC_CGM) for specifics on register implementation.

Configure the Auxillary Clock Selectors and Dividers before enabling the corresponding module. After every reset, first clear the Auxiliary Clock Mux selected source (MC_CGM_ACm_SC)and the division values (MC_CGM_ACm_DCp). And only then, select the required clock source and division values. Here m refers to (0..15) and p refers to (0..2). After each reset the Auxillary Clock Selectors should be switched twice.

Table 5-1. MC_CGM relationship to clocks

MC_CGM Registers	Register Description	Output Clock	CMU
MC_CGM_SC_DC0	System Clock Divider Configuration 0	SYS_CLK	CMU_4
MC_CGM_SC_DC1	System Clock Divider Configuration 1	CORE1_CLK	CMU_9
MC_CGM_SC_DC2	System Clock Divider Configuration 2	CORE2_CLK	CMU_10
MC_CGM_SC_DC3	System Clock Divider Configuration 3	PBRIDGE_0_CLK PBRIDGE_1_CLK	CMU_11
MC_CGM_SC_DC4	System Clock Divider Configuration 4	DMA_CLK	—
MC_CGM_SC_DC5	System Clock Divider Configuration 5	CORE0_CLK	CMU_1 (on z4 Checker)

Table continues on the next page...

Table 5-1. MC_CGM relationship to clocks (continued)

MC_CGM Registers	Register Description	Output Clock	CMU
MC_CGM_AC0_DC0 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	MC_CLK	CMU_0
MC_CGM_AC0_DC2 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	ADC_CLK	CMU_3 (on ADC_0)
MC_CGM_AC1_DC0 MC_CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	FR_CLK	—
MC_CGM_AC2_DC0 MC_CGM_AC2_SC	Aux Clock 2 Divider Configuration 0 Aux Clock 2 Select Control	CAN_CLK	CMU_2 (on CAN_0)
MC_CGM_AC3_SC	Aux Clock 3 Select Control	PLL0_CLKIN	—
MC_CGM_AC4_SC	Aux Clock 4 Select Control	PLL1_CLKIN	—
MC_CGM_AC7_DC MC_CGM_AC7_SC	Aux Clock 7 Divider Configuration 0 Aux Clock 7 Select Control	SPT_CLK	CMU_7
MC_CGM_AC8_DC MC_CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	LFAST_CLK	CMU_5
MC_CGM_AC9_DC	Aux Clock 9 Divider Configuration 0	CLK_OUT1	—
MC_CGM_AC10_DC MC_CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	ENET_CLK	—
MC_CGM_AC11_DC MC_CGM_AC11_SC	Aux Clock 11 Divider Configuration 0 Aux Clock 11 Select Control	ENET_TIME_CLK	—
MC_CGM_AC12_DC MC_CGM_AC12_SC	Aux Clock 12 Divider Configuration 0 Aux Clock 12 Select Control	SPI_CLK	CMU_8 (on SPI_1/ LIN_1)
MC_CGM_AC13_DC MC_CGM_AC13_SC	Aux Clock 13 Divider Configuration 0 Aux Clock 13 Select Control	LIN_CLK	
MC_CGM_AC14_DC	Aux Clock 14 Divider Configuration 0	CLK_OUT0	CMU_6

5.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in the following table. In addition, the system clocks are subject to the following limitations:

- TCK frequency should be less than half of SYS_CLK frequency. For example, if SYS_CLK is running on IRC, TCK frequency should be less than 8 MHz
- Core_1/Core_2 clock can always run in 2:1 or 1:1 with respect to SYS_CLK (Core_1/Core_2: SYS_CLK)
- DMA_CLK cannot be slower than AIPS clock
- DMA_CLK: SYS_CLK = 1:1
- SYS_CLK: PBRIDGE_n_CLK can only be 2:1
- SPT_CLK > SYS_CLK

The device has three frequency modes; Z7 cores will work on 240, 180 and 120 Mhz respectively in these modes. It is expected there may be performance scale-down in the frequency scaled-down mode (120 MHz). Accesses from Z7 cores and accesses from other masters to Z7 TCM will be impacted accordingly.

Table 5-2. Platform Clock Modes

	Z7 Cores	Z4 Core		
Mode	Core_1/Core_2	Core_0	AHB/SRAM/FLASH (SYS_CLK)	IPS (PBRIDGE_x_CLK)
4:2:2:1	240	120	120	60
4:4:2:1	180	180	90	45
2:2:2:1	120	120	120	60

NOTE

The user is responsible to verify that these values are not exceeded.

Each PBRIDGE (PBRIDGE_0, PBRIDGE_1) is capable of clocking the PBRIDGE interface of each peripheral slot individually at the XBAR frequency, or at the XBAR frequency divided by two.

In **2:2:2:1** mode, the user must program the source clock to be 120 MHz and not 240 MHz or higher for the traces to work. The divider values should be programmed accordingly.

Table 5-3. Maximum system level clock frequencies

System clock	Maximum frequency
CORE0_CLK	180 MHz plus modulation (+/-2%)
SYS_CLK	120 MHz plus modulation (+/-2%)
PBRIDGE_x_CLK	60 MHz plus modulation (maximum modulation expected is +/-2%)
CORE2_CLK	240 MHz plus modulation (+/-2%)
CORE1_CLK	240 MHz plus modulation (+/-2%)
DMA_CLK	120 MHz plus modulation (maximum modulation expected is +/-2%)
MC_CLK	160 MHz
ADC_CLK	80 MHz
SD_CLK	320 MHz
DAC_CLK	10 MHz
CTE_CLK	80 MHz
WGM_CLK	80 MHz
SPT_CLK	200 MHz

Table continues on the next page...

Table 5-3. Maximum system level clock frequencies (continued)

System clock	Maximum frequency
FR_CLK	40 MHz
CAN_CLK (FlexCAN)	40 MHz
LFAST_CLK	20 MHz
CLK_OUT0	60 MHz
LIN_CLK	80 MHz
LINIPS_CLK	40 MHz
CLK_OUT1	60 MHz
ENET_CLK	50 MHz
ENET_TIME_CLK	100 MHz
SDPLL80_CLK	80 MHz
SDPLL160_CLK	160 MHz
SDPLL320_CLK	320 MHz
XOSC_CLK	40 MHz
IRC_CLK	16 MHz
SPI_CLK	80 Mhz

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

5.3.1 Supported clocking configuration for magic carpet modes

The support clocking scenarios are listed below per MC_ME modes

Table 5-4. DRUN/RUNx/HALT0

SDPLL ON/OFF	PLL1 ON/OFF	PLL0 ON/OFF	XOSC ON/OFF	IRC ON/OFF	SYS_CLK SOURCE	Source to PLL0	Source to PLL1	_MC(CONFIG) values[1]
OFF	ON	ON	ON	ON	PLL1	XOSC	PLL0	0x001300F4
OFF	ON	ON	ON	ON	PLL1	XOSC	XOSC	0x001300F4
ON	ON	ON	ON	ON	PLL1	XOSC	PLL0	0x001301F4
ON	ON	ON	ON	ON	PLL1	XOSC	XOSC	0x001301F4
OFF	ON	OFF	ON	ON	PLL1	NA as PLL0 is OFF	XOSC	0x001300B4
ON	ON	OFF	ON	ON	PLL1	NA as PLL0 is OFF	XOSC	0x001301B4

1. _MC (CONFIG) value is actually the MC_ME_*_MC for the respective modes. Example MC_ME_DRUN_MC for DRUN mode.

Table 5-5. STOP0

SDPLL ON/OFF	PLL1 ON/OFF	PLL0 ON/OFF	XOSC ON/OFF	IRC ON/OFF	SYS_CLK SOURCE	Source to PLL0	Source to PLL1	_MC(CONFIG) values[1]
OFF	OFF	OFF	OFF	ON	IRC	NA as PLL0 is OFF	NA as PLL1 is OFF	0x00130010
OFF	OFF	OFF	ON	ON	IRC	NA as PLL0 is OFF	NA as PLL1 is OFF	0x00130010

1. MC_ME_STOP0_MC

NOTE

This does not reflect the configurations for self test modes

5.3.2 Aurora clock

The supported clock frequency of the Aurora clock input has a valid range of 625 MHz to 1.25 GHz. The NAL/NPC works at CORE0_CLK or SYS_CLK frequencies. The frequency of this clock must be > Aurora clock ÷ 10. For 1.25 GHz Aurora clock frequency, the SYS_CLK frequency must be ≥ 125 MHz.

Since the SYS_CLK is limited to 120 MHz, therefore, the Aurora clock is limited to 1.15GHz. When SYS_CLK is 90Mhz, the Aurora clock will be limited to 870 MHz (considering worst case setup and hold time).

NOTE

In case a mode transition is made into a mode such that the resultant SYS_CLK frequency is lower than the limit mentioned above, then the Aurora trace will get stalled. The connection with the debugger will need to be restarted after the SYS_CLK goes above the limit again.

During reset the SYS_CLK source is always IRC so this limitation also holds true, and the user will need to reconnect to the debugger after reset. If the reset is a software generated functional reset, then the user can avoid the process of reconnection by flushing out all traces from the chip before starting the reset event.

5.3.3 CSE clock

To meet SHE1.1 performance targets, configure Concurrent Secure Boot Mode through DCF Records (Refer to attached DCF Sheet) so that the CPU can configure the PLL to run the CSE at 120 MHz, instead of 16MHz/2 IRCOSC clock which is default clocking source.

5.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to one at power on reset.

NOTE

Do not access unimplemented address space for XOSC and IRCOSC register areas OR write software that is not dependent on receiving an error when access to unimplemented XOSC and IRCOSC space occurs.

5.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- SDPLL
- External Oscillator (XOSC)
- External Clock (EXTAL Bypass)
- 16 MHz Internal RC Oscillator (IRCOSC)

5.5.1 PLL

The PLL in the device is a Dual PLL which provides separate system and peripheral clocks as shown in the following figure. The PLLs are disabled after power on and must be enabled by software. The block diagram for the Dual PLL is shown in the following figure.

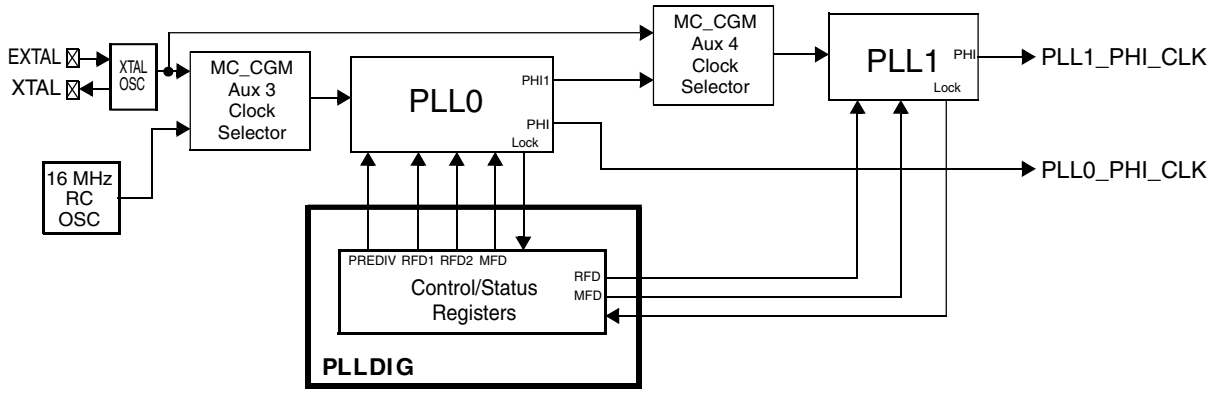


Figure 5-6. Dual PLL digital interface block diagram

5.5.1.1 PLL0 - base PLL (non-FM)

PLL0 is the primary PLL. This PLL is used to source a non-Frequency Modulated clock to the device modules and also the reference clock to PLL1.

5.5.1.1.1 PLL0 input clocks

The possible input clock sources for PLL0 are the XOSC_CLK and external reference clock. The external reference clock input is the EXTAL pin. AUX Clock Selector 3 selects which input clock will be used as the source for PLL0.

NOTE

IRC_CLK should not be used as reference clock of PLL in normal application. The Jitter and clock variation on PLL while running IRC will be too high for any useful application purposes.

5.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock (PLL0_PHI_CLK) drives the peripheral clock and the system clock when PLL1 is not locked or active. The PHI1 output provides one of the input references for PLL1.

5.5.1.2 PLL1 - FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) which is used to drive the system clock. PHI is the output of PLL1 which drives the System Clock and AUX Clock Selectors of the MC_CGM.

5.5.1.2.1 PLL1 input clocks

Either the XOSC output or the PHI1 signal from PLL0 may be selected (AUX Clock Selector 4 in the MC_CGM) as the source for PLL1.

5.5.1.2.2 PLL1 output clocks

The output clock from PLL1 is the PHI clock, which drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1_PHI_CLK frequency.

5.5.1.2.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the Dual PLL Digital Interface (PLLDIG) chapter.

5.5.1.3 SDPLL

SDPLL drives the System Clock and AUX Clock Selectors of the MC_CGM. SDPLL also drives SDADC and DAC. It is the primary source for MC_CLK. It is the primary source for MC_CLK. SDPLL should be enabled only after the configuring SPT_ACQ_GBL_CTRL_0[CSI_ADC_SEL] field. This is necessary to avoid glitches in the clock of SPT and MIPI modules.

5.5.1.4 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in the following table. All other register reset values are shown in the Dual PLL Digital Interface (PLLDIG) chapter.

Table 5-6. Dual PLLDIG register reset values

Offset (hex)	Register	Reset value (hex)
0000h	PLL0CR - PLL0 Control Register	0000_0000h
0008h	PLL0DV - PLL0 Divider Register	0000_0000h
0020h	PLL1CR - PLL1 Control Register	0000_0000h
0028h	PLL1DV - PLL1 Divider Register	0000_0000h

5.5.1.5 PLL initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC.

1. Configure PLL0 and related modules.
   a. With PLL0 disabled, program the PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is the 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select the XOSC as source.
   b. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD], and PLL0DV[RFDPHI].
   c. If desired, modify the AFE_OSCDLY[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).
   d. The XOSC frequency is compared to the IRCOSC source when the XOSC is turned on.
2. Turn on the XOSC and PLL0.
   a. Configure a mode configuration for turning on PLL0 and the XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<*mode*>_MC[XOSCON] = 1 and MC_ME_<*mode*>_MC[PLL0ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0 PHI output) as the sysclk.
   b. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.
3. Wait for the mode transition to complete.
   a. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:
      - The XOSC counter expires (if the new mode configuration changes MC_ME_<*mode*>_MC[XOSCON] = 1), and
      - PLL is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL0ON] = 1)
   b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].
4. Configure PLL1 and related modules.
   a. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 0 which selects PLL0_PHI1.
   b. Program PLL1DV[MFD] and PLL1DV[RFDPHI].

c. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

5. Turn on PLL1.
   a. Configure a mode configuration for turning on PLL1, and keeping the XOSC and PLL0 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<*mode*>_MC[XOSCON] = 1, MC_ME_<*mode*>_MC[PLL0ON] = 1 and MC_ME_<*mode*>_MC[PLL1ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.
   b. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if re-entering the same mode.
6. Wait for the mode transition to complete.
   a. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:
      - XOSC counter expires (if the new mode configuration changes XOSCON to 1), and
      - PLL1 is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL1ON] = 1).
   b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

NOTE

See the Mode Entry Module (MC_ME) chapter for details.

5.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also can be used as a clock source to the ADCs, timer, serial interfaces, LFAST, and as a system clock source. It is available for observation on either of the CLK_OUT*n* pins, and is used as a reference to calibrate the IRCOSC frequency.

The external oscillator of 40 MHz input frequency allows a crystal or external clock to be used as the reference clock for the microcontroller.

5.5.2.1 XOSC register write protection

The XOSC registers have write protection as shown in the Register Protection table.[1]

5.5.2.2 XOSC reset value

The following table shows the default reset value for the XOSC registers.

Table 5-7. XOSC register reset values

Offset	Register	Reset value
00h	Oscillator Control Register (AFE_OSCCTRL)	0000_0001h
04h	Oscillator Status Register (AFE_OSCSTS)	0000_0001h
08h	Oscillator Delay Register (AFE_OSCDLY)	0000_0080h

5.5.3 16 MHz internal RC oscillator (IRCOSC)

The microcontroller has a 16 MHz internal RC oscillator which is always enabled and can be used as the clock source for the PLLs. The IRCOSC is the default clock after reset.

5.6 Peripheral Clocks

Figure 5-4 shows the clock distribution to peripherals.

5.6.1 Enabling of peripherals and clock sources

Software should take care that the clock sources (system clocks and protocol clocks) selected for a peripheral are already enabled in the current mode before initiating the target mode transition to enable the peripheral. For example, in FlexCAN when XOSC is selected for the CAN engine clock source (CAN_CTRL1[CLKSRC]), before enabling FlexCAN peripheral clock the user must ensure XOSC is enabled in the current mode prior to the target mode transition that will enable the FlexCAN.

1. See Register Protection section in Device Configuration chapter for bit field details.

5.6.2 Disabling of peripherals and clock sources

Software should take care that the clock sources (system clocks and protocol clocks) selected for a peripheral are not disabled during any mode transition preceding a mode transition to disable the peripheral. For example, in FlexCAN when XOSC is selected for the CAN engine clock source (CAN_CTRL1[CLKSRC]), before disabling FlexCAN peripheral clock the user must ensure XOSC is enabled in the current mode prior to the target mode transition that will disable the FlexCAN. If need be, XOSC should be disabled only after that.

5.6.3 LFAST clocking

The device includes an LFAST module to support inter-processor communication. A single LFAST PLL, which requires a 20 MHz reference, supports high speed operation of the LFAST module. For low speed LFAST operation, the reference clock is used directly by the LFAST module.

The following figure shows the clock routing for the various LFAST modules on the device, including connections to the DRCLK_IO and DRCLK pins.

The LFAST PLL requires a 20 MHz reference. The source may be the PLL0_PHI_CLK,PLL1_PHI_CLK, IRC_CLK, SDPLL_CLK, the external oscillator (XOSC_CLK) or input from the external LFAST device via PAD_135 package pin. When this reference clock is generated internally, it can also be output on the PAD_135 pin. The PAD_135 connection is shown in Figure 5-7. Input/Output direction of PAD_135 pin is controlled by MSCR135 in the SIUL2.

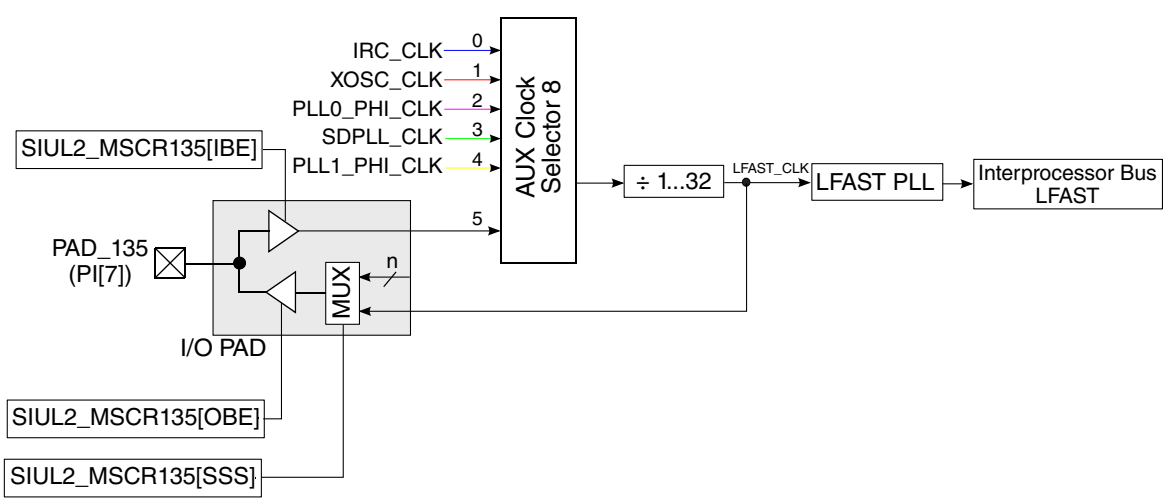


Figure 5-7. LFAST clocking

5.6.4 FlexRay clocking

The FlexRay protocol clock is sourced from either the non-FM PLL0_PHI_CLK, PLL1_PHI_CLK, XOSC_CLK, IRC_CLK, or SDPLL_CLK. When using the external oscillator clock, a 40 MHz crystal is required for proper operation. Clock selection is controlled by configuring the FR_MCR register, and the selection is output from the FlexRay block.

The clock mux is a glitchless mux and requires both the inputs to be on during clock switching.

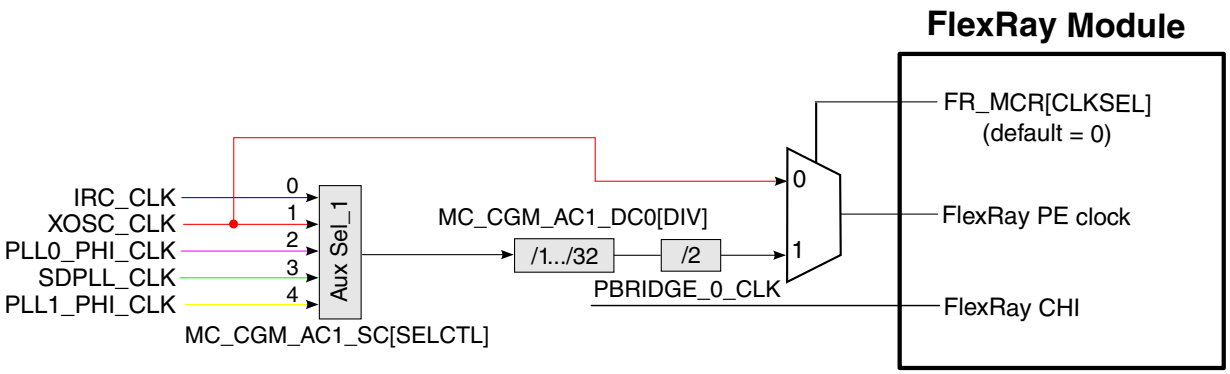


Figure 5-8. FlexRay clocking

5.6.5 FlexCAN clocking

As seen in Figure 5-9, the CAN_CLK is generated from either the XOSC, PLL0_PHI_CLK, IRC_CLK, SDPLL_CLK, or PLL1_PHI_CLK. The selected clock source can be divided by a value written to the MC_CGM_AC2_DC register in the MC_CGM. The output of this divider (CAN_CLK_OUT) drives one of two inputs of a multiplexer on the FlexCAN. The other input is driven by XOSC_CLK. The FlexCAN_CTRL1[CLK_SRC] bit selects whether the CAN_CLK_OUT or XOSC_CLK will drive the output of the multiplexer for the CAN_CLK(PE Clock/Protocol Clock).

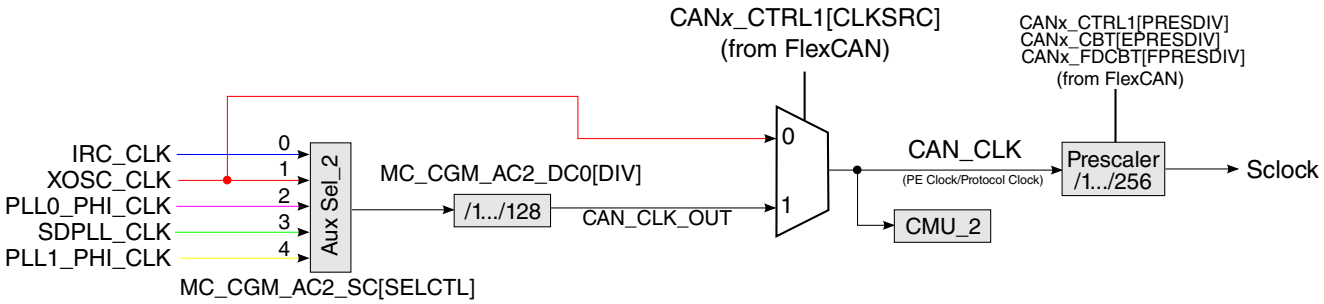


Figure 5-9. FlexCAN clocking

NOTE

FlexCAN modules should never be kept on in STOP mode. They must be disabled by configuring a Mode Entry Module Low-Power Peripheral Configuration Register (MC_ME_LP_PCn) for peripheral clock frozen in STOP mode and selecting this low-power configuration in the appropriate MC_ME Peripheral Control Register.

5.6.6 MIPICSI2 clocking

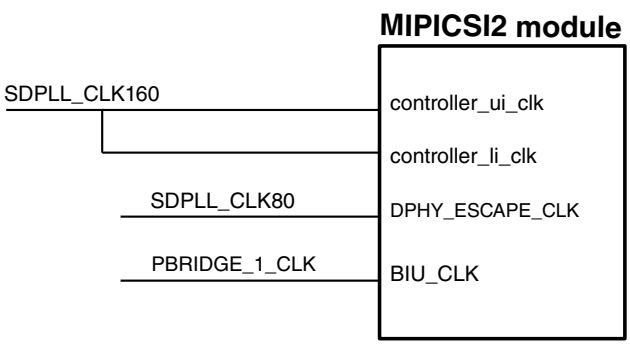


Figure 5-10. MIPICSI2 clocking

5.6.7 ENET clocking

The ENET's AHB interface clock is SYS_CLK. Its BIU clock is PBRIDGE_0_CLK.

The ENET protocol clock is sourced from either the PLL0_PHI_CLK, PLL1_PHI_CLK, XOSC_CLK, IRC_CLK, SDPLL_CLK, or from an external clock source as selected by the MC_CGM_AC10_SC register. These sources can be divided by a value written to the MC_CGM_AC10_DC register in the MC_CGM. The output of this divider is ENET_CLK.

The ENET time is sourced from either the PLL0_PHI_CLK, PLL1_PHI_CLK, XOSC_CLK, IRC_CLK, SDPLL_CLK, or from an external clock source as selected by the MC_CGM_AC11_SC register. These sources can be divided by a value written to the MC_CGM_AC11_DC register. The output of this divider is ENET_TIME_CLK.

NOTE

A ENET clock requirement is that the SYS_CLK frequency must be at least 2X the ENET_CLK. Eg for MII at 100 Mbps, the ENET_ClK is 25 MHz and therefore the SYS_CLK must be at least 50 MHz.

NOTE

RGMII mode operation is limited to **4:2:2:1** system clocking mode and needs 120 MHz SYS_CLK.

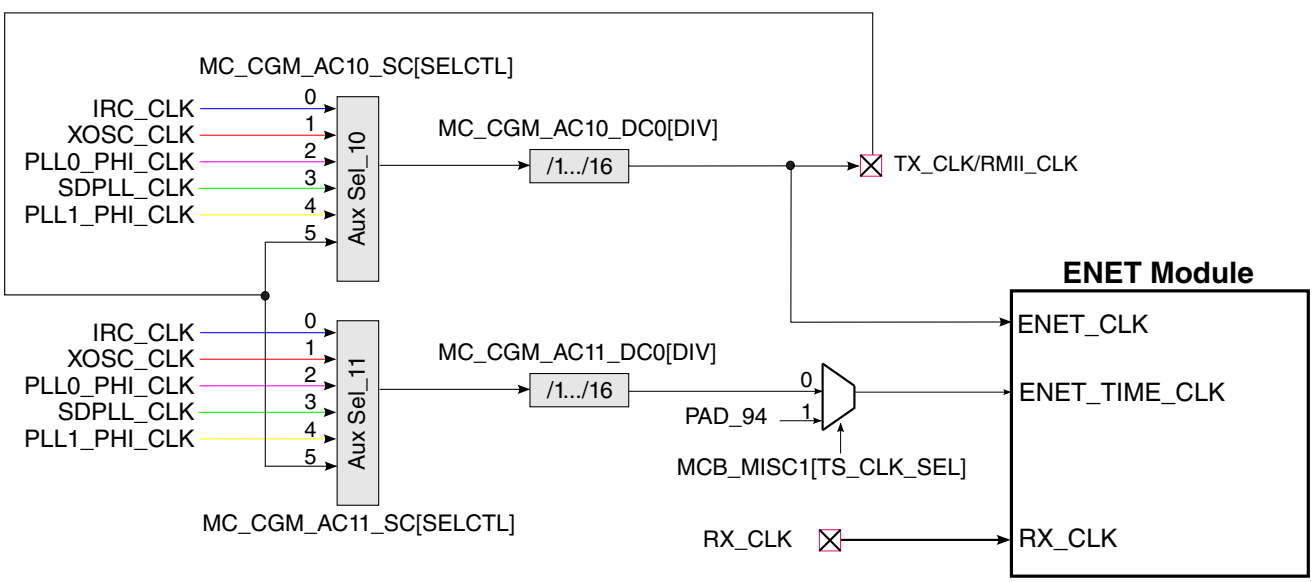


Figure 5-11. ENET clocking

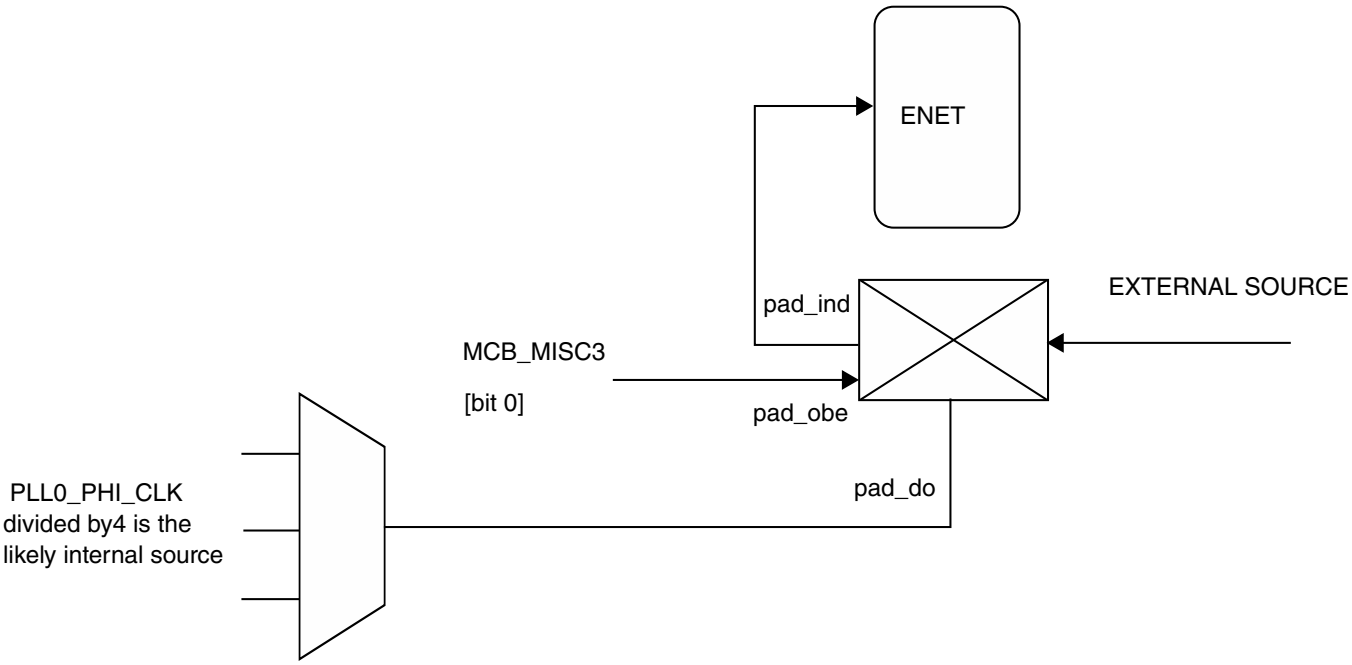


Figure 5-12. ENET RMII clocking

MCB_MISC3[RMII_MODE] is used to control the ENET clocking mode:

- 0: External clock is used as the source of RMII
- 1: This chip will provide RMII clock outside

A proper clock selection must be made on AUX10. The preferred clock is PLL0_PHI_CLK (200 MHz) divided by 4.

In RGMII Mode, reference clock for RGMII_TXC is driven from PAD_130.

5.6.8 SPT clocking

The SPT clock is sourced from either the PLL0_PHI_CLK, PLL1_PHI_CLK, XOSC_CLK, IRC_CLK, or SDPLL_CLK. The PLL0_PHI_CLK can be divided by a value written to the CGM_AC7_DC register in the MC_CGM. The output of this divider is SPT_CLK.

5.6.9 LINFlexD clocking

LINFlexD can select (using MC_CGM_AC13_SC) from one of seven possible clock sources for its LIN_CLK. LIN_CLK is then used to create the baud clock.

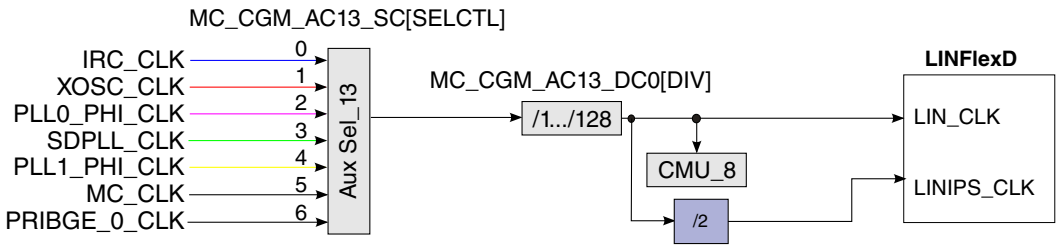


Figure 5-13. LINFLexD clocking

5.6.10 I²C clocking

On this chip, the PBRIDGE clock sources the I²C module clock, which in turn sources the I²C protocol clock. The IBCR[MDIS] field controls only whether the I²C core/protocol clock is enabled/disabled. It cannot disable the I²C module clock. Therefore, when the IBCR[MDIS] field's setting disables the I²C core/protocol clock:

- The I²C module itself is not entirely disabled.
- The field for the I²C module in the applicable MC_ME Peripheral Status register does not indicate that the module is "frozen."

5.7 Clock monitoring

For all safety critical clocks the microcontroller detects a missing clock or incorrect frequency. To achieve this clock monitoring units (CMUs) are used. Each CMU is programmed independently. The IRC_CLK and XOSC_CLK are used as the clock monitor references. Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

5.7.1 CMU configuration

This section explains the CMU configuration.

Figure 5-14 shows the block diagram for CMU_0 used for MC_CLK, XOSC and IRCOSC monitoring.

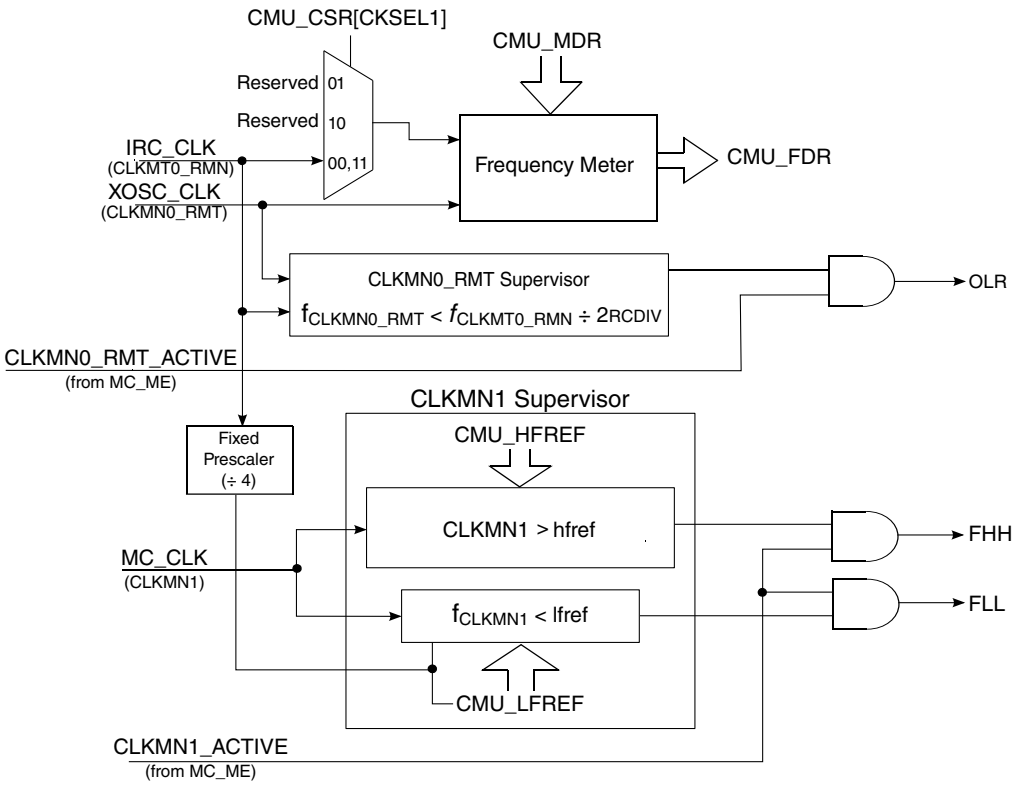


Figure 5-14. CMU_0 Block Diagram

NOTE

CLKMN0_RMT_ACTIVE refers to status (whether it is active or not) of XOSC_CLK. This can also be checked from the status in MC_ME_GS[S_XOSC].

CLKMN1_ACTIVE refers to the status (whether it is active or not) of MC_CLK. This can also be checked from the status in MC_ME_GS of the clock selected in CGM_AC0_SS for MC_CLK .

Figure 5-15 shows the block diagram for CMU_8 used for LINFlexD clock monitoring.

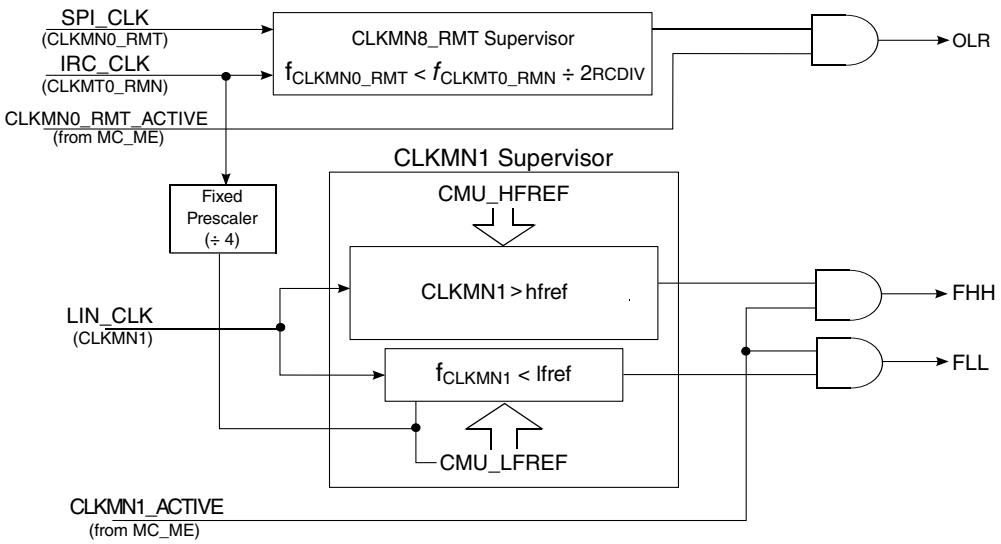


Figure 5-15. CMU_8 Block Diagram

NOTE

CLKMN0_RMT_ACTIVE refers to status (whether it is active or not) of SPI_CLK. This can also be checked from the status in MC_ME_GS of the clock selected in CGM_AC12_SS for SPI_CLK.

CLKMN1_ACTIVE refers to the status (whether it is active or not) of LIN_CLK. This can also be checked from the status in MC_ME_GS of the clock selected in CGM_AC13_SS for LIN_CLK.

Figure 5-16 shows the block diagram for CMU_[1:7,9:12].

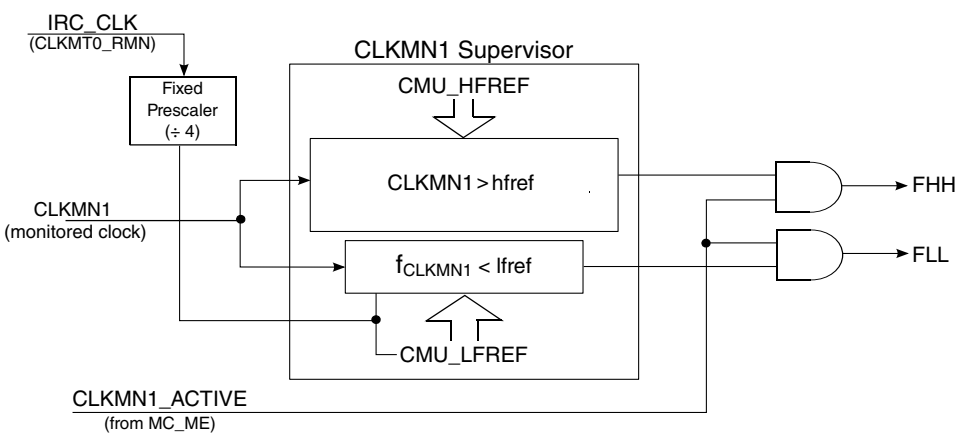


Figure 5-16. CMU_[1:7,9:12] Block Diagram

NOTE

CLKMN1_ACTIVE shows the status (whether it is active or not) of CLKMN1, the source of which comes from the MC_CGM and varies depending on the CMU. Table 5-1 shows the relevant source for CLKMN1 for each CMU

5.7.1.1 Clock input sources

Table 5-8 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRC_CLK is the reference clock for all clock monitors. Only CMU_0 implements the XOSC monitor. CMU_0 uses the IRC_CLK clock to measure if the XOSC_CLK is too low. CMU_0 can also be used to calibrate the IRC_CLK frequency using the XOSC_CLK. All other CMUs are configured identically.

Table 5-8. Clock input sources

Clock module	Monitored clock
CMU_0	XOSC_CLK, IRC_CLK, MC_CLK
CMU_1	CORE0_CLK
CMU_2	CAN_CLK
CMU_3	ADC_CLK
CMU_4	SYS_CLK
CMU_5	LFAST_CLK
CMU_6	CLK_OUT0
CMU_7	SPT_CLK
CMU_8	LIN_CLK, SPI_CLK
CMU_9	CORE1_CLK
CMU_10	CORE2_CLK
CMU_11	PBRIDGE_0_CLK, PBRIDGE_1_CLK
CMU_12[1]	SPT_ACQN_CLK

1. CMU12 monitored clock is dependent on SPT acquisition unit clock source. In SDADC-SPT use-case, input clock to CMU12 will be SDPLL_CLK80 and in MIPI-SPT use case, input clock to CMU12 will be SDPLL_CLK 160.

5.7.1.2 CMU registers and field availability

This table specifies which registers and fields are available on a given CMU for this device

Table 5-9. CMU registers availability

Address Offset	Register	CMU	Note
0000h	CMU_CSR	All	Only CMU_0 has the fields CMU_CSR[SFM] and CMU_CSR[CLSEL1]. Only CMU_0 and CMU_8 have the field CMU_CSR[RCDIV].
0004h	CMU_FDR	CMU_0	-
0008h	CMU_HFREFR	All	-
000Ch	CMU_LFREFR	All	-
0010h	CMU_ISR	All	CMU_ISR[OLRI] is in CMU_0 and CMU_8 only.
0014h	Reserved	-	-
0018h	CMU_MDR	CMU_0	-

5.7.2 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set, and an interrupt is generated, if enabled. Frequency limits are given in the device Data Sheet.

5.7.3 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU_0, using the XOSC_CLK as a reference. This allows for application trimming of the IRC_CLK frequency.

5.7.4 System clock monitors

A CMU is assigned to monitor the frequency of the checker core, both Peripheral bridges, motor control, and ADC clocks.

5.7.4.1 PLL0 Monitor

Software can program an upper and lower limit on the expected PLL0_PHI_CLK frequency. If the monitor is enabled, and the measured frequency is above or below the limits, a flag bit is set, and an interrupt is generated, if enabled. The default condition of the clock monitor is disabled.

5.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application.

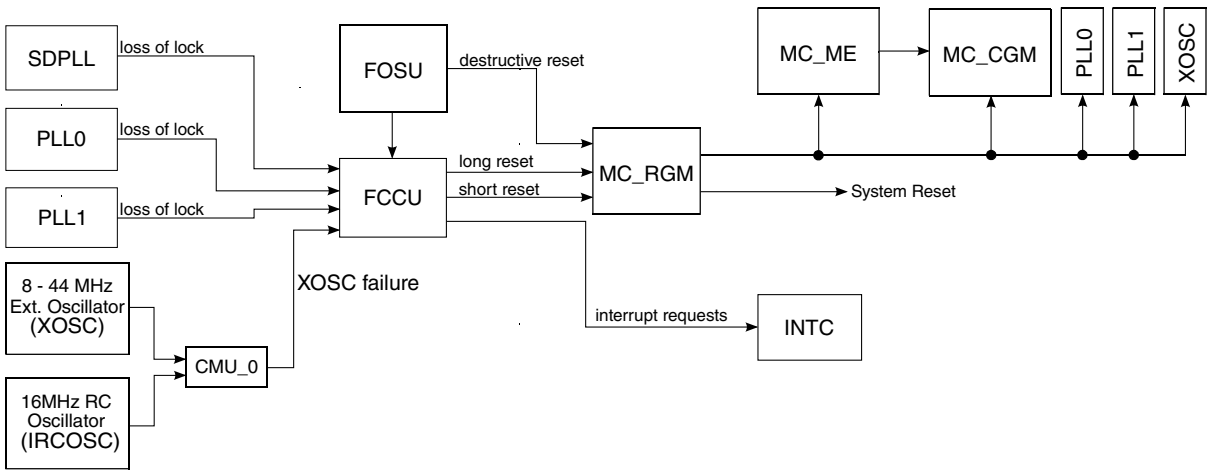


Figure 5-17. Loss of clock logic

5.8.1 Loss of PLL/XOSC_CLK

As shown in Figure 5-17, each loss of lock signal from each PLL and the XOSC_CLK failure signals are monitored by the FCCU. The FCCU can be programmed to generate a short or long reset sequence when a clock failure occurs, or simply to generate an interrupt. In the case of the interrupt, only the backup clock in the PLL can provide clocks to the system in order to perform a system clock switch. There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

If a long reset is selected the PLL, XOSC_CLK, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the IRC_CLK. If a short reset is selected, the clock configuration is maintained, but again, the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.

NOTE

There is no direct interrupt corresponding to PLL Loss of Lock, if an interrupt is desired then it must be configured as the relevant NCF (for PLL loss of lock) reaction via the FCCU after enabling the LOLIE bit.

5.8.2 Loss of IRC_CLK

The frequency of the IRC_CLK is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRC_CLK fails. Since the IRC_CLK is the boot and backup clock, a failure is a catastrophic failure.

5.9 Progressive clock switching

The Progressive Clock Frequency Switching (PCFS) blocks are shown in the system-level clock diagram in Figure 34-3. They are used for ramping the system and peripheral clocks. See the clock Generation Module (MC_CGM) chapter for more information on the PCFS.

Chapter 6
Reset Overview

6.1 Introduction

The chapter describes the global and local reset behavior of the device.

6.1.1 Global Reset

The global reset behavior refers to device reset behavior controlled by the MC_RGM which affects several modules of the device at the same time. The global reset is linked to the reset process described in Global Reset Process.

6.2 Global Reset Process

This section describes the global reset process. Throughout this section the term *global* is omitted.

6.2.1 Overview

The global reset process is a sequence of several reset phases. Each reset phase has a specific entry condition, a specific exit condition, and a specific device reset behavior, which is different among the reset phases. The reset phases are executed in a specific order, thus building the actual global reset process. There are several modules contributing to the global reset process. All modules of the device can be affected by the global reset process.

6.2.2 Reset Process Modules

All modules which can enter, gate, or control the reset process are listed in Table 6-1 together with their type of contribution. The types of contribution are described in Table 6-2.

Table 6-1. Reset Process Modules

Module	Contribution Type		
	event	gating	control
Reset Generation Module (MC_RGM)	v	v	v
Power Management Controller (PMC)	v	v	v
Mode Entry Module (MC_ME)	v		
System Status and Configuration Module (SSCM)		v	
Self-Test Control Unit (STCU2)	v		
Fault Collection and Control Unit (FCCU)	v	v	
JTAG Controller (JTAGC)	v		
RADAR Analog Front End (AFE)	v		
Embedded Flash Memory (c55fmc)		v	
RC Oscillator (IRCOSC)		v	

Table 6-2. Module Contribution Type

Contribution Type	Description
event	The module may generate events which may trigger the entry into a reset state, if the event configuration in the MC_RGM allows this.
gating	The module can prevent the exit of a reset state if it has not reached a certain internal state.
control	The module controls the sequence of the reset process.

6.2.2.1 Reset Generation Module (MC_RGM)

The MC_RGM is one of the two controlling module for the reset process. The MC_RGM can receive events from other modules and starts the reset process based on the configuration for these events. The event configuration defines the reset phase at which start the reset process starts. The MC_RGM controls the reset process sequence based on the gating information it receives from the gating modules.

6.2.2.2 Power Management Controller (PMC)

The PMC provides two types of signals for the reset process control.

As long as the supply voltages are not reliable, the PMC provides signals to put the whole device into a power on reset state.

As long as the supply voltages are reliable but above or below certain operational limits, the PMC provides signals to allow the device to enter the reset process.

6.2.2.3 Mode Entry Module (MC_ME)

The MC_ME can generate events to enter the reset process. These events can be triggered only by application software.

6.2.2.4 System Status and Configuration Module (SSCM)

The SSCM is started automatically during a specific phase of the reset process. The SSCM fetches DCF records from the flash memory and distributes the read values via an internal DCF bus to the related modules. As long as the SSCM is still fetching DCF records, this specific reset phases can not be exited.

6.2.2.5 Self-Test Control Unit (STCU2)

The STCU2 controls the execution of the start-up and shut-down selftest. Depending on its configuration the STCU2 is started when PHASE3[DEST] is left or when software triggers the selftest. The STCU2 generates a functional event when the self testing has been completed.

In case, SSCM security reset gets generated while STCU2 self test (offline or online) is running, the SSCM security reset shall get masked. After STCU2 self-test run is over, long functional reset is issued to all LBIST partitions. The SSCM security reset (which is a source of destructive reset) is generated again during UTEST scanning. Hence, the system gets the destructive reset which would trigger offline STCU2 self-test (if not bypassed through DCF record) and this cycle repeats.

6.2.2.6 Fault Collection and Control Unit (FCCU)

The FCCU generates reset events and gates the reset process while it is configuring itself based on the DCF record content.

6.2.3 Reset Process Sequences

This section specifies the reset processes sequences. An overview of the reset process sequences is given in the following figure. When the reset process has reached the IDLE state, the device is operational.

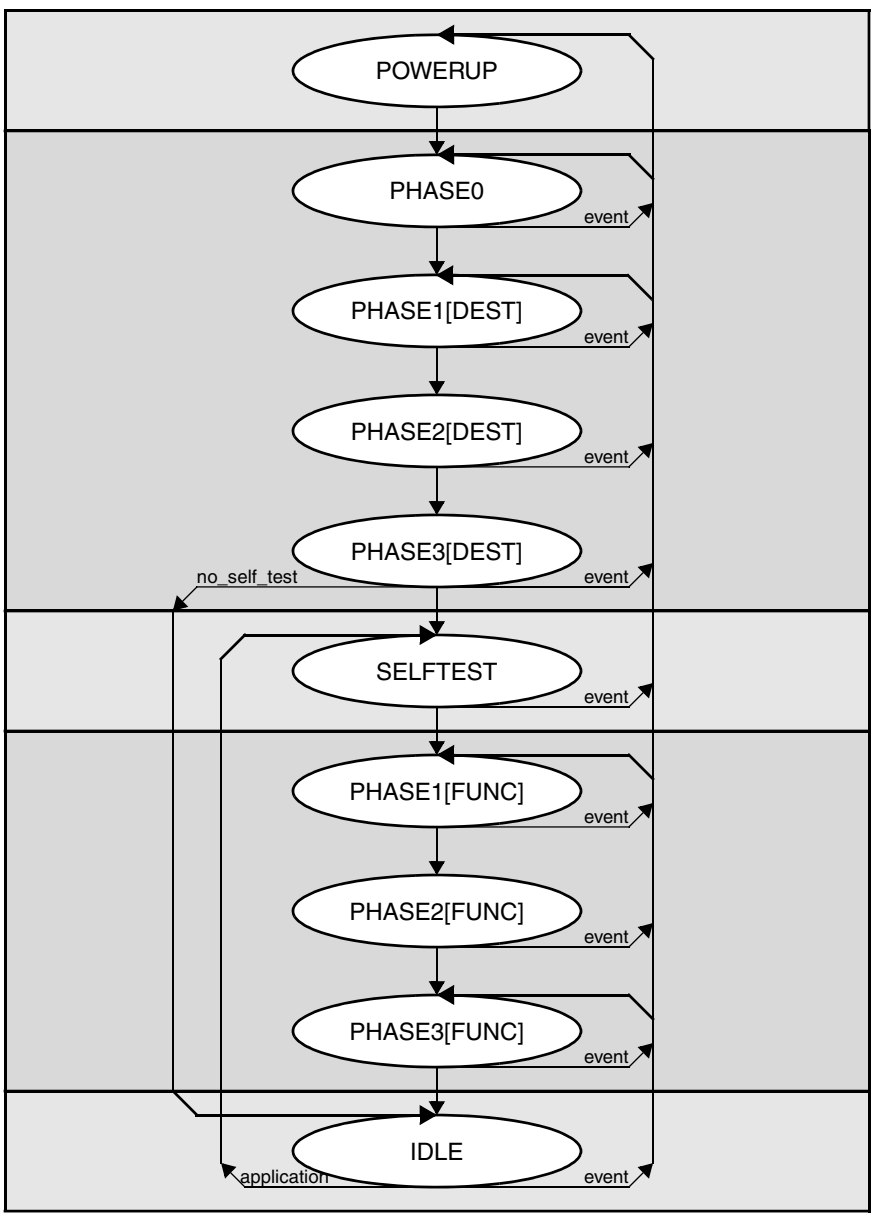


Figure 6-1. Reset Process Sequences

6.2.4 Reset Process State Transitions

This sections specifies the transitions between the reset states in the reset process. For each state, the state entry, the state execution, and the state exit is described.

6.2.4.1 POWERUP

6.2.4.1.1 POWERUP State Entry

This state is entered from any state if at least one of the following events appear

- Power is applied to the unpowered device.
- PMC has detected a POR condition.
- VREG_POR_B pin is driven low

6.2.4.1.2 POWERUP State Execution

During this state, the PMC is waiting for stable power supply.

6.2.4.1.3 POWERUP State Duration

There are no bounds on the duration of this state.

6.2.4.1.4 POWERUP State Exit

Table 6-3. POWERUP Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	(1) The VREG_POR_B pin is high	IRCOSC MC_RGM

6.2.4.2 PHASE0

6.2.4.2.1 PHASE0 State Entry

This state can be entered from

- all other reset states when the state exit condition is present.

6.2.4.2.2 PHASE0 State Execution

All trimming bits are reset to their default values.

6.2.4.2.3 PHASE0 State Duration

The maximum duration of this state is dependent on all voltages achieving the right voltage range.

6.2.4.2.4 PHASE0 State Exit

Table 6-4. PHASE0 Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE1[DEST]	(1) The MC_RGM indicates that all sources of enabled destructive reset events are deasserted, and (2) the IRCOSC module indicates the generation of a stable clock, and (3) the PMC module indicates that all supplies are in the operational range, that is, the PMC module has deasserted the following LVD-HVD signals (AFE regulator LVDs are by default masked. They can be unmasked using Multipurpose registers.)	

6.2.4.3 PHASE1[DEST]

6.2.4.3.1 PHASE1[DEST] State Entry

This state can be entered from state

- PHASE0,
- PHASE1[FUNC],
- PHASE2[FUNC],
- PHASE3[FUNC],
- IDLE.

6.2.4.3.2 PHASE1[DEST] State Execution

There are no state changes in any module, only a counter in the MC_RGM is running.

6.2.4.3.3 PHASE1[DEST] State Duration

The duration of this state is 512 IRCOSC clock cycles.

6.2.4.3.4 PHASE1[DEST] State Exit

Table 6-5. PHASE1[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE2[DEST]	512 IRCOSC clocks passed by after state entry.	FLASH - starts initialization sequence FCCU - enters configuration state in preparation for receiving DCF record content STCU - waits for configuration via DCF records

6.2.4.4 PHASE2[DEST]

6.2.4.4.1 PHASE2[DEST] State Entry

This state can be entered from state

- PHASE1[DEST].

6.2.4.4.2 PHASE2[DEST] State Execution

FLASH executes initialization sequence and is ready for array accesses.

6.2.4.4.3 PHASE2[DEST] State Exit

Table 6-6. PHASE2[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE3[DEST]	The FLASH module indicates the availability of array accesses.	SSCM

Table 6-6. PHASE2[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
		- starts device configuration and boot location search

6.2.4.5 PHASE3[DEST]

6.2.4.5.1 PHASE3[DEST] State Entry

This state can be entered only from state

- PHASE2[DEST].

6.2.4.5.2 PHASE3[DEST] State Execution

SSCM runs device configuration and boot location search.

The FCCU executes its configuration sequence.

The PMC adjusts its LVD and HVD levels according to the DCF record content, adjusts its output to the full voltage level, and executes its self test.

6.2.4.5.3 PHASE3[DEST] State Exit

Table 6-7. PHASE3[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
SELFTEST	(1) The SSCM module indicates device configuration is done, and (2) the FCCU module indicates it has completed its configuration sequence, and (3) the PMC module indicates it has finished its internal selftest, and (4) the STCU module is configured to execute a start-up self test.	BIST on - FCCU - CORE - PERIPHERALS
IDLE	(1) The SSCM module indicates device configuration is done,	CORE ALL OTHER PERIPHERALS

Table 6-7. PHASE3[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
	and (2) the FCCU module has completed its configuration sequence, and (3) the PMC module indicates it has finished its internal selftest, and (4) the FLASH module indicates the availability of array accesses.	

6.2.4.6 SELFTEST

6.2.4.6.1 SELFTEST State Entry

This state can be entered only from state

- PHASE3[DEST],
- IDLE.

6.2.4.6.2 SELFTEST State Execution

The start-up self test is executed as configured in the STCU by the DCF record content.

6.2.4.6.3 SELFTEST State Exit

Table 6-8. SELFTEST Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[FUNC]	STCU module indicates end of self test.	

6.2.4.7 PHASE1[FUNC]

6.2.4.7.1 PHASE1[FUNC] State Entry

This state can be entered from states

- SELFTEST,
- PHASE3[FUNC],
- IDLE.

6.2.4.7.2 PHASE1[FUNC] State Execution

There are no state changes in any module, only a counter in the MC_RGM is running.

6.2.4.7.3 PHASE1[FUNC] State Duration

The duration of this state is 512 IRCOSC clock cycles.

6.2.4.7.4 PHASE1[FUNC] State Exit

Table 6-9. PHASE1[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE2[FUNC]	(1) The MC_RGM indicates that all sources of enabled long functional reset events are deasserted, and (2) 512 IRCOSC clocks passed by after state entry.	FLASH - starts initialization sequence FCCU - enters configuration state in preparation for receiving DCF record content if PHASE1[FUNC] was entered on the exit of the SELFTEST state

6.2.4.8 PHASE2[FUNC]

6.2.4.8.1 PHASE2[FUNC] State Entry

This state can be entered from state

- PHASE1[FUNC].

6.2.4.8.2 PHASE2[FUNC] State Execution

FLASH executes initialization sequence and is ready for array accesses.

6.2.4.8.3 PHASE2[FUNC] State Exit

Table 6-10. PHASE2[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the sources of an enabled long functional reset event is asserted.	
PHASE3[FUNC]	The FLASH module indicates the availability of array accesses.	SSCM - starts device configuration

6.2.4.9 PHASE3[FUNC]

6.2.4.9.1 PHASE3[FUNC] State Entry

This state can be entered from states

- PHASE2[FUNC]
- IDLE.

6.2.4.9.2 PHASE3[FUNC] State Execution

SSCM runs device configuration, if state was entered from PHASE2[FUNC].

The FCCU executes its configuration sequence if PHASE3[FUNC] was entered via the SELFTEST state.

6.2.4.9.3 PHASE3[FUNC] State Duration

The minimum duration of this state is 40 IRC cycles.

6.2.4.9.4 PHASE3[FUNC] State Exit

Table 6-11. PHASE3[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	

Table continues on the next page...

Table 6-11. PHASE3[FUNC] Next State Table (continued)

Next State	Condition	Process/Module Started at Exit
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the sources of an enabled long functional reset event is asserted.	
IDLE	(1) The SSCM module indicates device configuration is done, and (2) the FCCU module has completed its configuration sequence, and (3) the PMC module indicates it has finished its internal selftest, and (4) the FLASH module indicates the availability of array accesses.	CORE ALL OTHER PERIPHERALS

6.2.4.10 IDLE

6.2.4.10.1 IDLE State Entry

The IDLE state can be entered only from states

- PHASE3[DEST]
- PHASE3[FUNC].

6.2.4.10.2 IDLE State Execution

During the IDLE state, the device is fully operational.

6.2.4.10.3 IDLE State Duration

There are no bounds on the duration of this state.

6.2.4.10.4 IDLE State Exit

The IDLE state next state table is shown in Table 6-12 the following table.

Table 6-12. IDLE Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the source of an enabled long functional reset event is asserted.	
PHASE3[FUNC]	The MC_RGM has detected that the source of an enabled short functional reset event is asserted.	
SELFTEST	The application has triggered the start of the shut-down selftest.	

6.2.5 Module Status During Reset Process

This section specifies in Table 6-13 the status of the device modules during the reset process. The status abbreviation explained in Table 6-14.

Table 6-13. Module Status during reset states [SCG_0180]

	Global Reset State									
Module	**POWER UP**	**PHASE0**	**PHASE 1[DEST]**	**PHASE 2[DEST]**	**PHASE 3[DEST]**	**SELFTEST**	**PHASE 1[FUNC]**	**PHASE 2[FUNC]**	**PHASE 3[FUNC]**	**IDLE**
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
IRCOSC	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
FLASH	RST	RST	RST	ON	ON	ON	RST	ON	ON	ON
SSCM	RST	RST	RST	RST	ON	ON	RST	RST	ON	ON
FCCU	RST	RST	RST	ON	ON	BIST	ON	ON	ON	ON
STCU	RST	RST	RST	ON	ON	ON	ON	ON	ON	ON
CORE	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON
OTHERS	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON

Table 6-14. Module Status Glossary

Module Status	Status Description
RST	Module is held in reset by the global reset process
BIST	Module is being tested or held in a non-functional state during self test execution as controlled by the STCU.
ON	Module is functional.

Chapter 7
System Boot

7.1 Overview

The device supports two types of boot schemes:

- Normal Boot
- Secure Boot

7.2 Normal boot

7.2.1 Entering boot modes

The device detects the boot mode based on SSCM_STATUS[BMODE]. BMODE is decided by LifeCycle and status of FAB (Force Alternate Boot Mode) and ABS (Alternate Boot Selector) pins (see the following table).

1. To boot either from FlexCAN or LINFlex, the device must be forced into an Alternate Boot Loader Mode via the FAB pin which must be asserted before initiating the reset sequence. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pin. Boot from FlexCAN and LinFLEX is only available when LifeCycle is MCU_PROD or CUST_DEL.
2. If FAB is not asserted and LifeCycle is MCU_PROD or CUST_DEL, the device boots from the first flash-memory sector which contains a valid boot signature.
3. If LifeCycle is OEM_PROD or IN_FIELD, then FAB and ABS pins are ignored and device always boots from Flash.
4. If no flash memory sector contains a valid boot signature, the device will go into static mode.
5. If Life cycle (LC) = Failure Analysis (FA), then device will go into static mode.

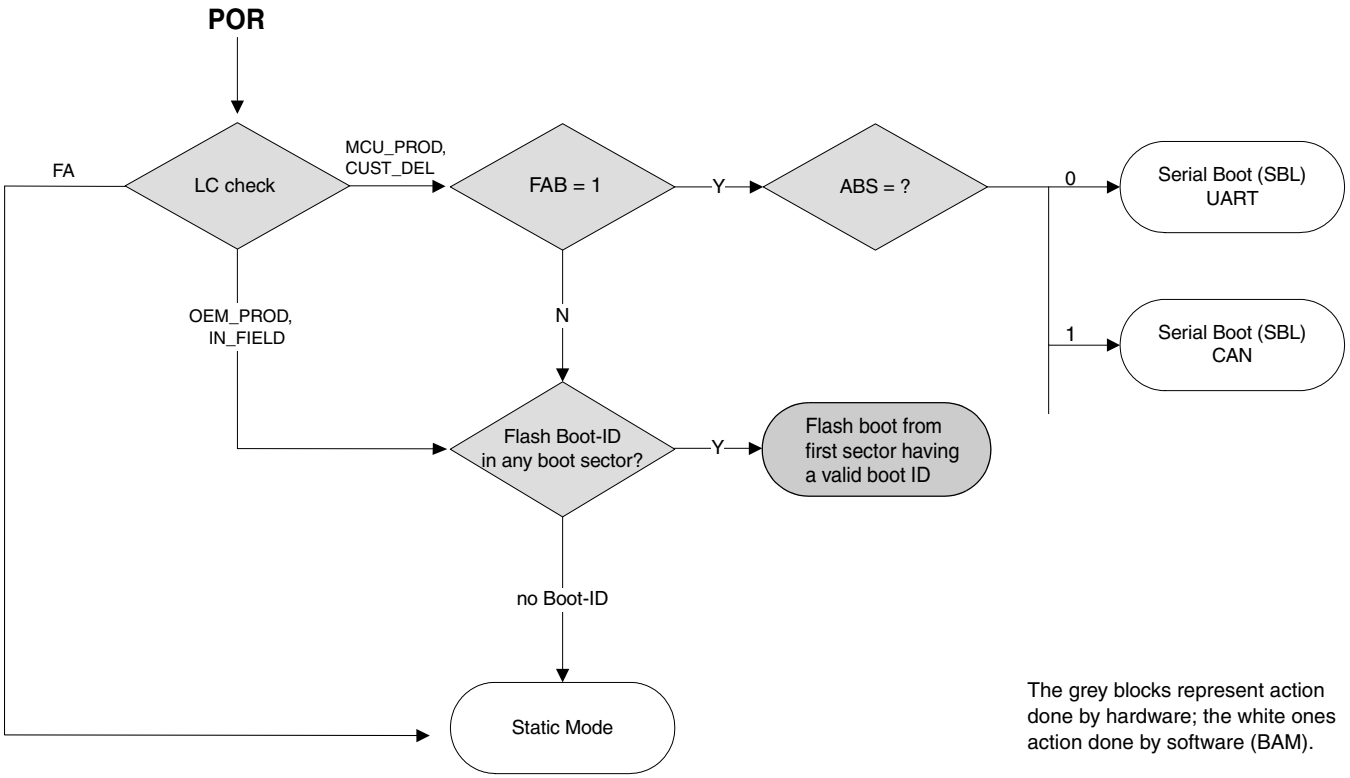


Figure 7-1. Boot mode selection

Table 7-1. Hardware configuration to select boot mode

LC	FAB pin	ABS pin	SSCM_STATUS[BMODE]	Functionality
MCU_PROD, CUST_DEL	0	x	Single Chip/Flash Boot	Flash Boot
MCU_PROD, CUST_DEL	1	0	SCI Serial Boot loader	UART Boot (BAM)
MCU_PROD, CUST_DEL	1	1	CAN Serial Boot Loader	CAN Boot (BAM)
OEM_PROD, IN_FIELD	x	x	Single Chip/Flash Boot	Flash Boot
FA	x	x	-	Static Mode

7.2.2 Boot Assist Module (BAM)

The Boot Assist Module (BAM) is a block of read-only memory containing VLE code which is executed according to the boot mode of the device. The code stored in the BAM is not executed when booting in Single Chip mode, except when entering the "Static mode" in case no valid boot-able section has been found, in case lifecycle is Failure Analysis, if invalid address or password is downloaded through serial boot or in case of any communication error over serial interface.

The BAM downloads code into internal SRAM through the following serial protocols and executes it afterwards:

- FlexCAN (CAN_0)
- LINFlex-UART (LINFlex_1)

Depending on the selected boot mode (see Entering boot modes), any download is performed with a fixed baud rate.

NOTE

'Differential XOSC' is not supported when booting from BAM in serial mode.

7.2.3 Features

The BAM provides the following features:

- 64-bit public password protection for serial boot mode
- Serial boot loads the application boot code from a FlexCAN or LINFlex bus into internal SRAM
- Configurable erroneous BAM accesses inhibit

7.2.4 Memory map

The BAM code resides in 8 KB of ROM mapped from address 0xFFFF_C000. The address space and memory used by the BAM is shown in the following table.

Table 7-2. BAM memory organization

Entity	Address
BAM entry point	0xFFFF_C000
RAM area used by the BAM code (do not use)	0x4000_0000–0x4000_00FF
Downloaded code base address	0x4000_0100
BAM exit address	0xFFFF_COCE

The RAM location where to download the code can be any 8 byte-aligned location in the SRAM starting from the address 0x4000_0100.

Caution

Do not use the RAM area used by the BAM code as indicated in the preceding table.

7.2.5 Reset value after BAM execution

BAM uses various modules for its operation and this causes change in reset values of few registers. Impacted registers are listed in below table.

Table 7-3. Reset value after BAM execution

Module	Offset	Register name	Value	Comments
CAN	0x44	CRC Register	undefined	CRC of last packet after CAN download
CGM	0x7E4	CGM_SC_SS	0x0002_0000	Clock switch after request from MC_ME completed
MC_ME	0xC	MC_ME_IS	0x0000_0001	Mode transtion completion Status in BAM
	0x18	MC_ME_DMTS	0x3000_0000	Changed due to DRUN to DRUN mode Transtion in BAM
LINFlex	0x8	LINFlexD_SR	0x0000_0040	Last Recive Data on LIN/UART Download

7.2.6 Functional description

7.2.6.1 Boot through BAM

7.2.6.1.1 Executing BAM

Single chip boot mode (selecting the first bootable Flash sector) is managed by hardware and BAM doesn't participate in it.

BAM is executed in any of the following cases:

- Serial boot mode has been selected by FAB pin;
- Hardware hasn't found a valid Boot ID in any flash memory boot locations;
- If lifecycle is Failure Analysis.

If one of these conditions is true, the device fetches code at location 0xFFFF_C000 and the BAM application starts.

7.2.6.1.2 BAM software flow

The following figure describes the BAM logic flow.

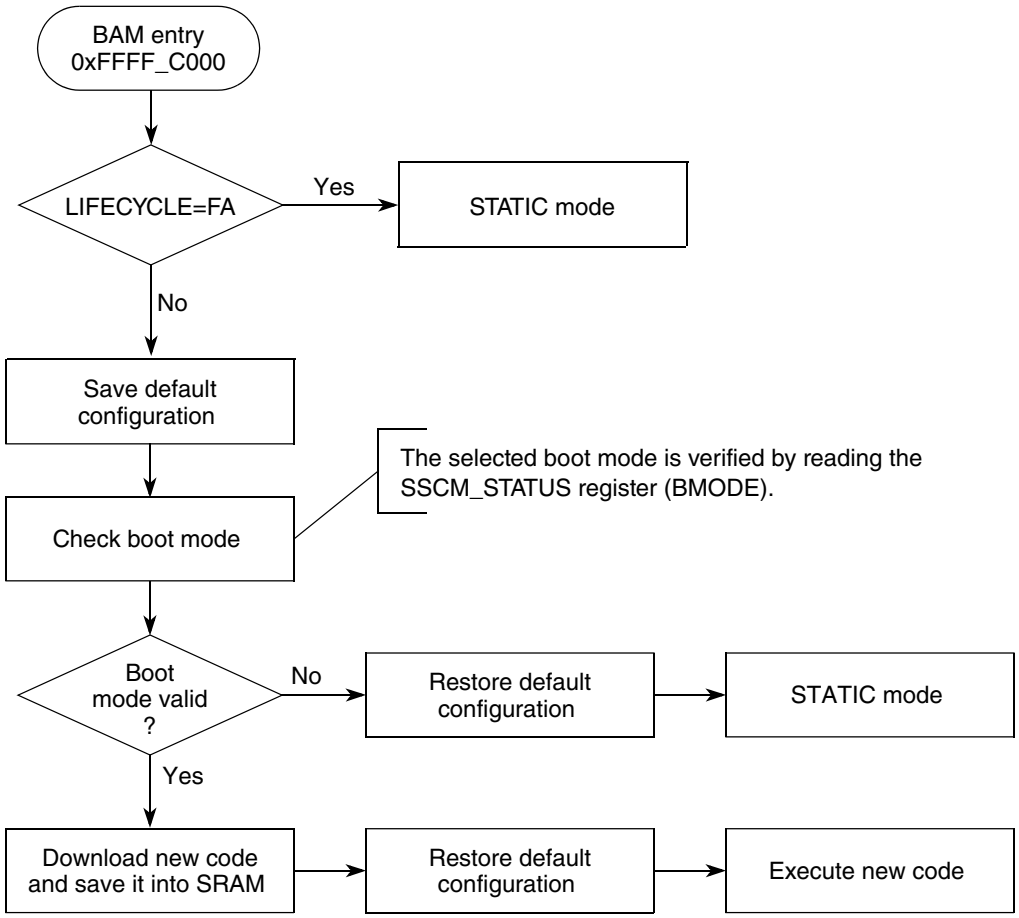


Figure 7-2. BAM logic flow

The first action is to save the initial device configuration. In this way it is possible to restore the initial configuration after downloading the new code but before executing it. This allows the new code to be executed as the device was just coming out of reset.

The SSCM_STATUS[BMODE] field indicates which boot has to be executed (see the following table).

If BMODE field shows the reserved values, the boot mode is not considered valid and the BAM pushes the device into static mode.

In all other cases data is downloaded in serial boot mode and saved into the correct SRAM location.

Table 7-4. Fields of SSCM STATUS register read by BAM to detect the chosen boot mode

Field	Description
BMODE	BMODE Device Boot Mode. 001 FlexCAN (FlexCAN_0) Serial Boot Loader 010 LINFlex-UART (LINFlex_1) Serial Boot Loader other values are reserved

Then, the initial device configuration is restored and the code jumps to the address provided for the downloaded code. At this point BAM has just finished its task.

If there is any error (that is, communication error, wrong boot mode selected, etc.), BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low power mode SAFE and the processor executes a wait instruction. It is needed if the device can not boot in the mode which was selected. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module MC_ME Module is "DRUN".

7.2.6.1.3 BAM resources

BAM uses/initializes the following MCU resources:

- MC_ME and MC_CGM modules to initialize mode and clock sources
- CAN_0, LINFlex_1 and corresponding pins when performing serial boot mode
- SWT is disabled in case of a serial boot mode or when entering static mode
- SSCM to check the boot mode and lifecycle (see Figure 7-2)
- External oscillator (XOSC)
- AFE for enabling external oscillator (XOSC)

The majority of the initial configuration is restored before executing the downloaded code.

The frequency of the external oscillator defines the baud rate for serial interfaces used to download the user application. For a LINFlexD transmission, the selected baud rate is f_{XOSC} / 833. For a FlexCAN transmission, the selected baud rate is f_{XOSC} / 40.

7.2.6.1.4 Download and execute the new code

From high level perspective, the download protocol follows these steps:

1. Transmit 64-bit password.
2. Transmit start address, size of downloded code in bytes, and VLE bit.
3. Transmit download data.
4. Execute code from start address.

Each step must be completed before the next step starts.

The communication is done in half duplex manner, any transmission from host is followed by the MCU transmission:

- Host sends data to MCU and starts waiting

- MCU echoes to host the data received
- Host verifies if echoes are correct
  - if data is correct, the host can continue to send data
  - if data is not correct, the host stops to transmit and MCU must be reset.

All multi-byte data structures are sent with MSB first.

A more detailed description of these steps follows.

NOTE

There must be no stringent timeout condition while the host waits for the echoed data from the MCU; especially after step 2. After receiving the actual size of the data to be transmitted, the BAM must initialize the memory receiving the transmitted data with 64 bit writes to avoid ECC errors due to uninitialized memory.

7.2.6.1.5 Download 64-bit password and password check

The first 64-bits received represent the password. This password is compared with PUBLIC password 0xFEEDFACECAFEBEEF. If password matches, then download continues otherwise BAM puts device is SAFE mode.

7.2.6.1.6 Download start address, VLE bit and code size

The next 8 bytes received by the MCU contain a 32-bit Start Address, the VLE mode bit and a 31-bit code length as shown in the following figure.

The Variable Length Instruction (VLE) bit is used to indicate the instruction set for which the code has been compiled. The BAM supports the download of VLE code.

The Start Address defines the location at which the received data will be stored and location at which the MCU will branch after the download is complete. The two LSB bits of the Start Address are ignored by the BAM program, such that the loaded code should be 32-bit word aligned.

The Code Length defines the number of data bytes to be loaded.

NOTE

BAM code allows code length upto 150KB. Start address should not be beyond 0x40100000.

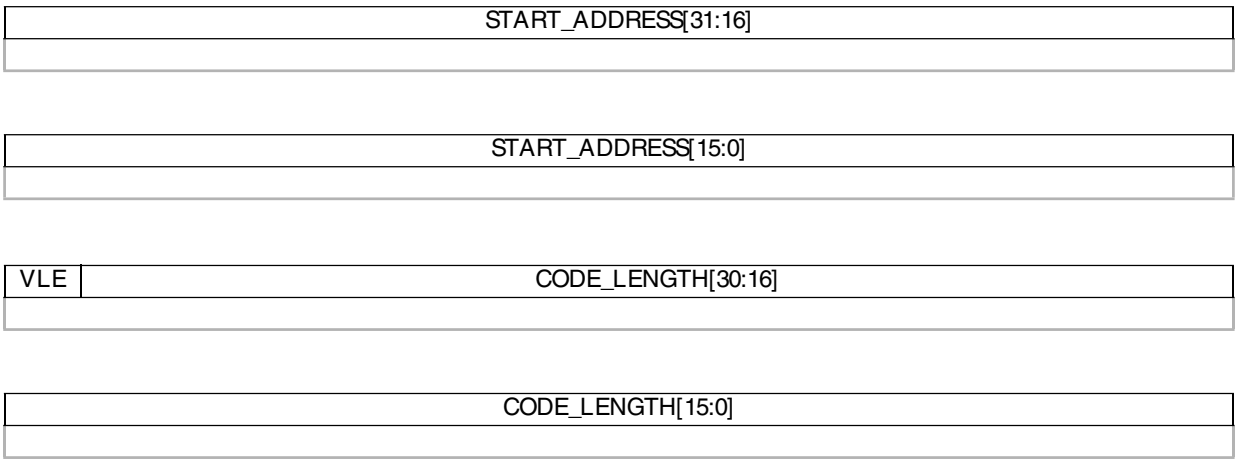


Figure 7-3. Start address, VLE bit, and download size in bytes

7.2.6.1.7 Download data

Each byte of data received is stored into device's SRAM, starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until the number of bytes of data received matches the number of bytes specified in the previous protocol step.

Since the SRAM is protected by a 64-bit wide Error Correction Code (ECC), the BAM performs a series of 64 bit writes to initialize the SRAM are required for the downloaded data. The actual writes are performed in chunks of 32-bit writes. If the last byte received does not fall onto a 32-bit boundary, the BAM fills it with 0 bytes.

Finally a "dummy" word (0x0000_0000) is written to avoid a possible ECC error during core prefetch.

7.2.6.1.8 Execute code

The BAM program waits for the last echo message transmission being completed.

Then it restores the initial MCU configuration and jumps to the code loaded at Start Address which was received in step 2 of the protocol.

At this point BAM has finished its tasks and MCU is controlled by new code executing from SRAM.

7.2.6.2 UART Boot

7.2.6.2.1 Configuration

Boot using the UART protocol is implemented by LINFlex_1 module. Pins used are:

- LINFlex_TX corresponds to pin PF[14]
- LINFlex_RX corresponds to pin PF[15]

The LINFlexD controller is configured to operate at a baud rate of f_{XOSC} / 833, using 8 bit data frame without parity bit and 1 stop bit.

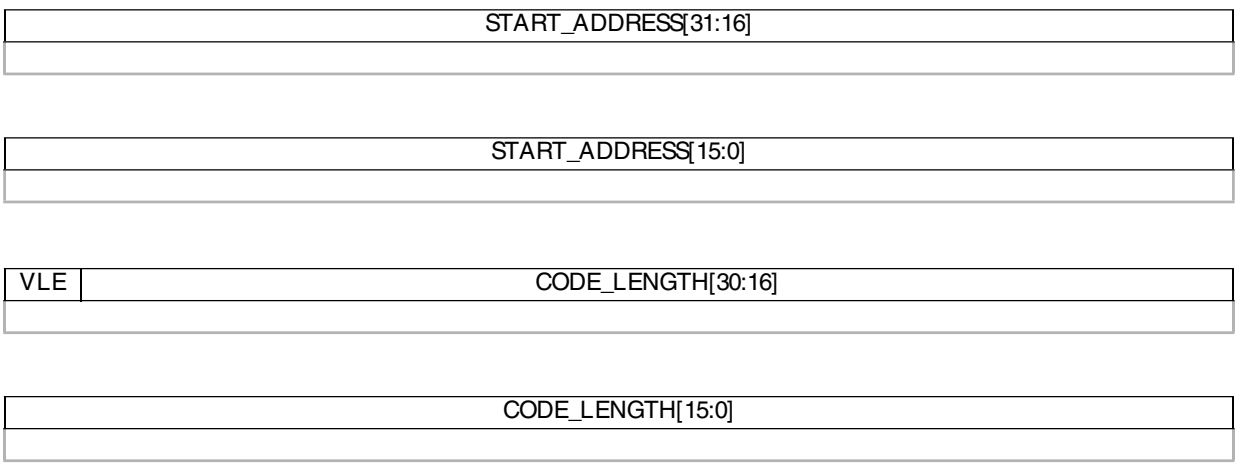


Figure 7-4. BAM LINFlex data frame

7.2.6.2.2 Protocol

The following table summarizes the protocol and BAM action during this boot mode.

Table 7-5. UART boot mode download protocol

Protocol step	Host sent message	BAM response message	Action
1	64-bit password (MSB first)	64-bit password	Password checked for validity and compared against stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31-bit number of bytes (MSB first)	VLE bit + 31-bit number of bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	8 bits of raw binary data	4 x 8 bits of data are packed into 32-bit words. These words are saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
5	none	none	Branch to downloaded code

7.2.6.3 CAN Boot

7.2.6.3.1 Configuration

Boot using the CAN protocol is implemented by FlexCAN_0 module. Pins used are:

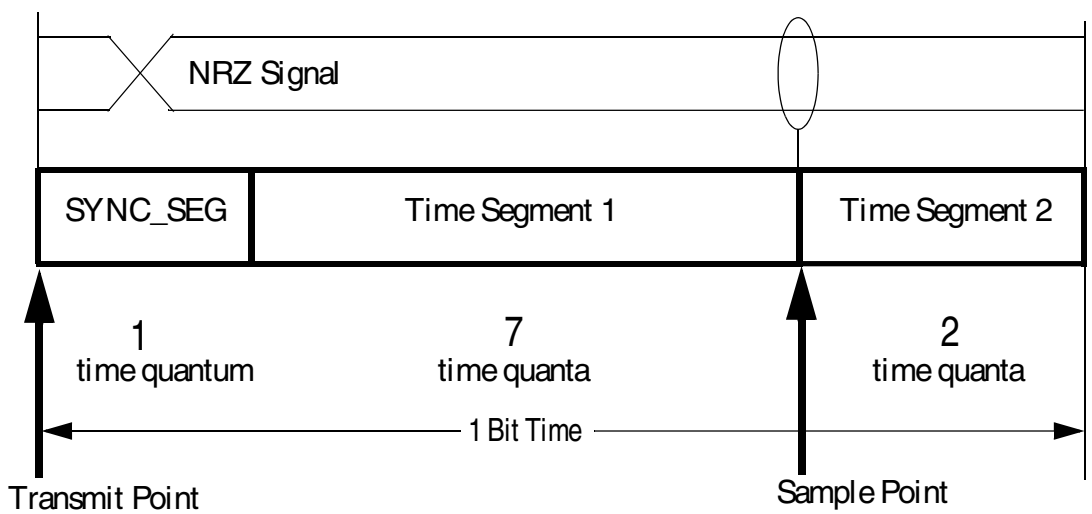
- CAN_TX corresponds to pin B[0]
- CAN_RX corresponds to pin B[1].

Boot from FlexCAN uses the system clock driven by the external oscillator.

The FlexCAN controller is configured to operate at a baud rate = system clock frequency/40.

It uses the standard 11 bit identifier format detailed in FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in the following figure.



1 time Quantum = 4 system clock periods

Figure 7-5. FlexCAN bit timing

7.2.6.3.2 Protocol

The following table summarizes the protocol and BAM action during this boot mode. All data is transmitted byte wise.

Table 7-6. FlexCAN boot mode download protocol

Protocol step	Host sent message	BAM response message	Action
1	FlexCAN ID 0x011+ 64-bit password	FlexCAN ID 0x001+ 64-bit password	Password checked for validity and compared against stored password.
2	FlexCAN ID 0x012+ 32-bit store address+ VLE bit+ 31-bit number of bytes	FlexCAN ID 0x002+ 32-bit store address+ VLE bit+ 31-bit number of bytes	Load address is stored for future use. Size of download are stored for future use. Verify VLE bit.
3	FlexCAN ID 0x013+ 8 to 64 bits of raw binary data	FlexCAN ID 0x003+ 8 to 64 bits of raw binary data	4 x 8 bits of data are packed into 32-bit words. These words are saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
4	none	none	Branch to downloaded code

7.2.6.4 Inhibiting BAM operation

Under certain circumstances, you may want to inhibit BAM operation. To do this, set the RAE bit in the SSCM_ERROR register. Any attempt to access the memory range occupied by the BAM will then result in an access error.

7.2.6.5 Interrupt

No interrupts are generated or are enabled by the BAM.

7.3 Secure boot

7.3.1 Overview

CSE supports secure boot execution.The secure boot process starts after a power-on-reset, a destructive or a functional reset if it is enabled through the DCF. For details, refer the DCF tables attached to this document as an Excel file. Secure Boot Command is issued through hardware after Phase3 Reset de-assertion.

After leaving the reset phase, either a concurrent or sequential secure boot can be executed, depending on the configuration through the DCF.

Secure boot is blocked in following cases:

1. Lifecycle is Failure Analysis
2. Serial Boot Mode is selected by asserting the FAB Pin.

The start address of secure boot verification is same as the address pointed by RCHW scanning of a valid boot Flash sector. The size of the code is configured by a DCF record in the UTEST flash sector. Size equal to zero shall be invalid.

The device supports two types of secure Boot mode:

1. Concurrent Secure Boot Mode
2. Sequential Secure Boot Mode

7.3.1.1 Concurrent Secure Boot Mode

When configured for concurrent secure boot, the cores execute code from the flash while the hardware triggers the secure boot of the CSE. Using concurrent secure boot enables a significantly faster verification time, due to the fact that the CPU can configure the PLL to run the CSE at up to 120MHz, instead of from the 16MHZ/2 IRCOSC clock. The CPU must be sure to suspend any CSE operations currently in progress, especially when changing the flash configuration, to avoid any errant calculations.

7.3.1.2 Sequential secure boot mode

When configured for sequential secure boot, the hardware triggers the CSE secure boot while the cores are held in reset. The core's reset are released once the secure boot execution is finished, regardless of whether the test passes for fails. The default system clock is (16 MHz)/2 IRCOSC and since CSE does not have the capability to configure the PLL nor the system clock, sequential secure boot is relatively slower.

In sequential secure boot mode, if CSE hangs (due to any reason), system would get out of reset due to internal CSE timeout but the CSE_SR[BSY] would remain set indicating unsuccessful secure boot. Software could take appropriate action to handle such scenario.

7.3.1.2.1 CSE Local RAM

The device supports 8 Kb of local secure RAM. The firmware is loaded from the secure flash into this RAM and it will be executed from there.

7.3.1.2.2 Chain of Trust

The verification can be performed with a single step over the entire software image or in several steps over partial software sections. The verification of the entire software image requires only a single CMAC, however since run time for a large software image can be high, security related functionality may not be available until the verification is completed. Also, this way of verification requires a continuous stored and monolithic software image. For improved device availability, shortened response time, the verification can be executed in several steps. This is also required if the application code is divided into several sections of non-contiguous addresses or is split into parts which are programmed by different parties. Additionally, multi-step verification enables to assign individual keys to certain code sections. If verification uses multiple steps over several code sections, a chain of trust must be established. In such a case, the verification must start with the first executed code and must not have gaps and span the whole application code. If verification is performed in several steps, only the first section, typically the boot loader, is verified with BOOT_MAC and all subsequent sections require their own CMAC. The CSE must be triggered by VERIFY_MAC command for all subsequent sections except the first one.

Chapter 8
Interrupt and DMA Overview

8.1 INTC configurable parameters

The INTC on this chip supports 3 processors, as shown in the following table:

Table 8-1. INTC processor number and corresponding core

INTC processor number	Corresponding CPU core
0	e200z420n3 (Core_0)
1	e200z7260n3 (Core_1)
2	e200z7260n3 (Core_2)

8.2 INTC interrupt sources

NOTE

For "Interrupt sources mapping", refer the device interrupt sources tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click on it.

8.3 INTC monitors

There are maximum 3 monitors per core.

8.4 DMAMUX sources

DMAMUX channel mapping: See the device DMAMUX sources tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click on it.

Chapter 9
Functional Safety

9.1 Introduction

The chip is developed according to ISO 26262 and has an integrated safety concept targeting safety-related systems requiring high safety integrity levels. In order to support the integration of the chip into safety-related systems, the following documentation is available:

- Reference Manual - Describes the S32R274 functionality
- Data Sheet (S32R274DS) - Describes the S32R274 operating conditions
- Safety Manual (S32R274SM) - Describes the S32R274 safety concept and possible safety mechanisms (integrated in S32R274, system level hardware or system level software), as well as measures to reduce dependent failures
- Dynamic FMEDA - Inductive analysis enabling customization of system level safety mechanisms, including the resulting safety metrics for ISO 26262 (SPFM, LFM and PMHF) and IEC 61508 (SFF and Beta IC Factor)
- FMEDA Report - Describes the FMEDA methodology and safety mechanisms supported in the FMEDA, including source of failure rates, failure modes and assumptions made during the analysis.

The FMEDA and FMEDA report are available upon request. The chip is a SafeAssure solution, for further information regarding functional safety at NXP, visit www.nxp.com/safeassure.

Chapter 10
Embedded memories

10.1 Overview

The embedded memory architecture for this microcontroller includes:

- On-board SRAM, including system SRAM, local data memory for each processor core, and on-board system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Embedded memories in the peripherals:
  - FlexRay
  - eDMA
  - Signal Processing Toolbox (SPT)
- PRAM_XBAR

10.2 System SRAM

This microcontroller includes up to 1.5 MB general-purpose on-chip ECC SRAM. Each of the 8 banks of SRAM can be independently configured for either 0- or 1-wait state read operation latency.

10.2.1 SRAM Controller (PRAMC)

This microcontroller's 8 banks of system SRAM interface to the Data Crossbar Switch (XBAR) via a quad-ported Platform RAM Controller (PRAMC). Internally, the quad-ported RAM controller consists of 8 independently configurable RAM controllers (one for each bank of system SRAM) connected to an internal Crossbar Switch. This is shown in Figure 10-1. Note that the configuration of the internal Crossbar Switch cannot be changed but it includes a Crossbar Integrity Checker (XBIC) that provides an error

detection and reporting capability and an error injection feature that is useful for system development and debugging. The XBIC appears in the System Memory Map as XBIC_2 (PRAM). See the Crossbar Integrity Checker (XBIC) chapter for details.

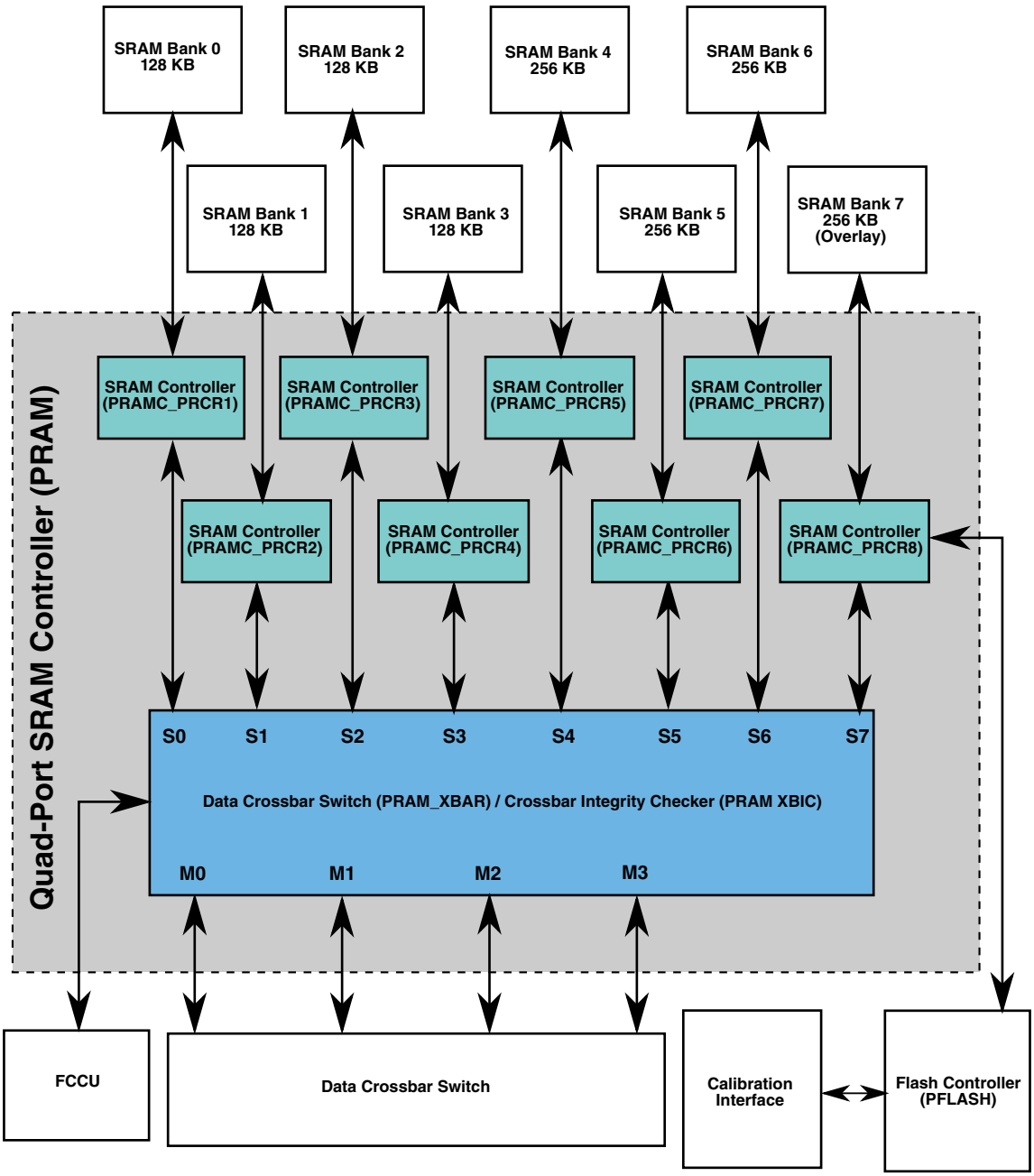


Figure 10-1. System SRAM architecture

10.2.2 Overlay SRAM

As shown in Figure 10-1, SRAM Bank 7 is available for use as *overlay SRAM*. The overlay SRAM feature included in this device is a feature used during calibration and debug activities. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables calibration of constant data without requiring additional external RAMs and calibration memory interfaces.

NOTE

1. Overlay SRAM is normally used for system development purposes only.
2. Timing for calibration accesses to a particular overlay SRAM are the same as access to the underlying flash if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The overlay function is implemented as follows using registers in the flash memory controller.

1. Using PFLASH Calibration Region Descriptors (PFCRDn) define one or more (up to 8) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

See the PFlash calibration remap support for details on memory overlay (remapping).

10.2.3 Processor core local SRAM

This microcontroller includes 64 KB ECC local data SRAM for each processor core to provide enhanced performance. Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices. See the System Memory Map for details.

10.2.4 Signal Processing Toolbox (SPT) memory

Signal Processing Toolbox (SPT) has 96 KB parity protected SRAM and Twiddle RAM. Twidlle RAM uses a bit line multiplexing of 4, separating logical neighbor bits by 3 physical bits of other logical words. SPT RAM uses a bit line multiplexing of 4, separating logical neighbor bit by 3 physical bits of other logical words.

10.3 PRAM_XBAR

10.3.1 Overview

Inside SRAM controller is a XBAR similar to the system XBARs with 4 masters (the 4 ports of the SRAM controller) and 8 slaves (the 8 SRAM memory banks). Configuration is similar to the system XBARs.

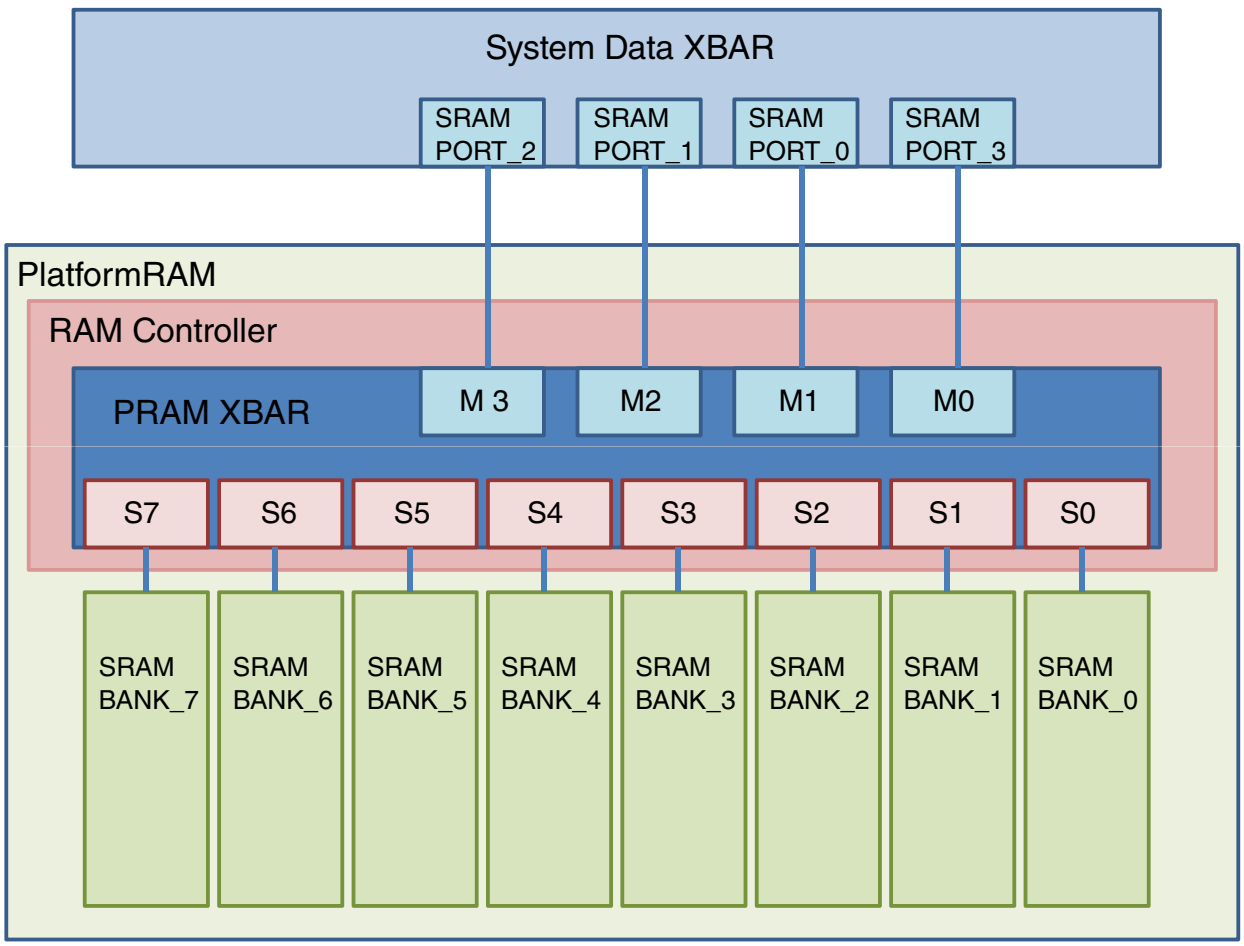


Figure 10-2. PRAM_XBAR block diagram

10.3.2 PRAM_XBAR master assignments

This device contains four masters connected to PRAM_XBAR. The master assignments are shown in the table below:

Table 10-1. PRAM_XBAR master assignments

Master module	Physical master port number
SRAM PORT_3	0
SRAM PORT_0	1
SRAM PORT_1	2
SRAM PORT_2	3

10.3.3 PRAM_XBAR slave assignments

This device contains eight slaves connected to PRAM_XBAR. The slave assignments are shown in the table below:

Table 10-2. PRAM_XBAR slave assignments

Slave Module	Physical slave number
SRAM BANK_0	0
SRAM BANK_1	1
SRAM BANK_2	2
SRAM BANK_3	3
SRAM BANK_4	4
SRAM BANK_5	5
SRAM BANK_6	6
SRAM BANK_7	7

10.3.4 PRAM_XBAR registers

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if a unimplemented location is accessed within the crossbar switch.

The CRS*n* and PSR*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to learn will terminate with an error.

PRAM_XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Register Slave (PRAM_XBAR_PRS0)	32	R/W	0000_3210h	10.3.4.1/ 213
10	Control Register (PRAM_XBAR_CRS0)	32	R/W	000F_0000h	10.3.4.2/ 214
100	Priority Register Slave (PRAM_XBAR_PRS1)	32	R/W	0000_3210h	10.3.4.1/ 213
110	Control Register (PRAM_XBAR_CRS1)	32	R/W	000F_0000h	10.3.4.2/ 214
200	Priority Register Slave (PRAM_XBAR_PRS2)	32	R/W	0000_3210h	10.3.4.1/ 213
210	Control Register (PRAM_XBAR_CRS2)	32	R/W	000F_0000h	10.3.4.2/ 214
300	Priority Register Slave (PRAM_XBAR_PRS3)	32	R/W	0000_3210h	10.3.4.1/ 213
310	Control Register (PRAM_XBAR_CRS3)	32	R/W	000F_0000h	10.3.4.2/ 214
400	Priority Register Slave (PRAM_XBAR_PRS4)	32	R/W	0000_3210h	10.3.4.1/ 213
410	Control Register (PRAM_XBAR_CRS4)	32	R/W	000F_0000h	10.3.4.2/ 214
500	Priority Register Slave (PRAM_XBAR_PRS5)	32	R/W	0000_3210h	10.3.4.1/ 213
510	Control Register (PRAM_XBAR_CRS5)	32	R/W	000F_0000h	10.3.4.2/ 214
600	Priority Register Slave (PRAM_XBAR_PRS6)	32	R/W	0000_3210h	10.3.4.1/ 213
610	Control Register (PRAM_XBAR_CRS6)	32	R/W	000F_0000h	10.3.4.2/ 214
700	Priority Register Slave (PRAM_XBAR_PRS7)	32	R/W	0000_3210h	10.3.4.1/ 213
710	Control Register (PRAM_XBAR_CRS7)	32	R/W	000F_0000h	10.3.4.2/ 214

10.3.4.1 Priority Register Slave (PRAM_XBAR_PRS*n*)

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

PRAM_XBAR_PRS*n* field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 M3	Master 3 Priority Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port 001 This master has level 2 priority when accessing the slave port 010 This master has level 3 priority when accessing the slave port 011 This master has level 4 priority when accessing the slave port
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 Priority Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port 001 This master has level 2 priority when accessing the slave port 010 This master has level 3 priority when accessing the slave port 011 This master has level 4 priority when accessing the slave port
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port 001 This master has level 2 priority when accessing the slave port 010 This master has level 3 priority when accessing the slave port 011 This master has level 4 priority when accessing the slave port
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority

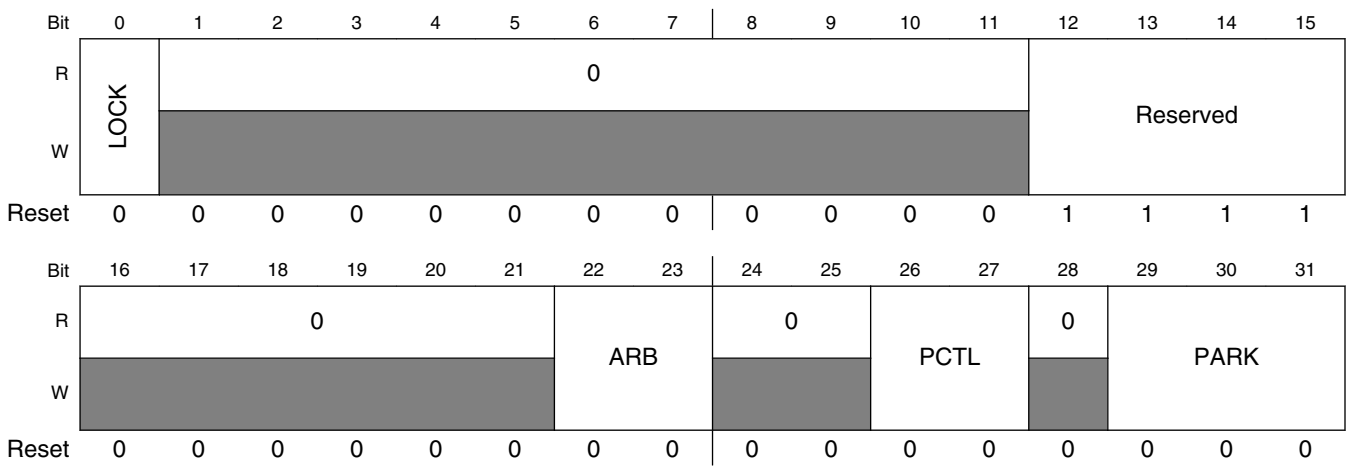
Table continues on the next page...

PRAM_XBAR_PRS*n* field descriptions (continued)

Field	Description
	Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port 001 This master has level 2 priority when accessing the slave port 010 This master has level 3 priority when accessing the slave port 011 This master has level 4 priority when accessing the slave port

10.3.4.2 Control Register (PRAM_XBAR_CRS*n*)

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d



PRAM_XBAR_CRS*n* field descriptions

Field	Description
0 LOCK	Locks the slave port's CRSn and PRSn registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writeable. 1 The slave port's registers are read-only and cannot be written. Attempted writes have no effect on the registers and result in a bus error response.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 Reserved	This field is reserved.
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin, or rotating, priority 10 Reserved 11 Reserved

Table continues on the next page...

PRAM_XBAR_CRS*n* field descriptions (continued)

Field	Description
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated. However, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Reserved 101 Reserved 110 Reserved 111 Reserved

10.4 Flash memory

This microcontroller includes up to 2 MB general-purpose on-chip flash memory (includes 64 KB EEPROM emulation) with the following features:

- Three ports (one per CPU) shared between Code and Data Flash with a 4x256 bit buffer for the Code and Data Flash including pre-fetch function
- Single and double bit error visibility are supported
- Support for reading-while-writing (RWW) for accesses to different partitions.
- Test information is stored in a non-volatile One Time Programmable (OTP) UTest block

NOTE

Secure flash blocks associated to CSE are available as data flash blocks only if CSE is disabled.

The included flash memory consists of a flash memory controller and a flash memory array module containing multiple blocks of non-volatile memory. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device's crossbar switch. The following figure shows the memory architecture.

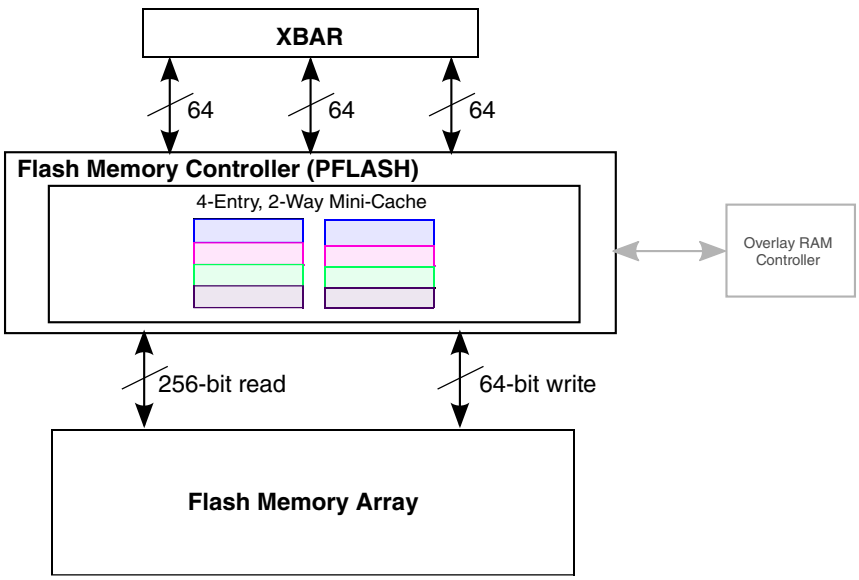


Figure 10-3. Device flash memory block diagram

The flash memory controller (PFLASH) occupies 3 slave ports (one per processor core) on the instruction crossbar switch. Flash memory configuration, performance tuning of flash memory accesses, and overlay RAM configuration are implemented using the flash memory controller's configuration registers. See the Flash Memory Controller chapter for details.

10.4.1 Flash memory controller

The flash controller acts as an interface between the system bus and the flash array and, when flash overlay SRAM is used, serves as the interface to the overlay RAM.

The flash controller contains a 4-entry, 2-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed flash array access.

The flash memory controller contains configuration registers which manage flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the flash. See the Flash Memory Controller chapter for details.

During Read While Write (RWW) accesses the flash memory controller may report the following non-critical faults to the FCCU (Fault inputs):

- NCF[43], which indicates that during an EDC after ECC check, the flash memory controller detected an error in the address ECC manipulation logic
- NCF[44], which indicates the flash memory controller detected a transaction monitor mismatch when compared to flash safety feedback output

The errors occur when a read operation is attempted on either a block currently being programmed or erased, or any block within the same flash memory partition. The error is also reported by the C55FMC flash memory in the C55FMC_MCR[RWE] bit.

10.4.2 UTest memory space

See 'UTEST Memory Map and DCF Sheet' attached as an Excel file with this document.

10.4.3 Chip-specific flash memory configuration

10.4.3.1 Flash memory controller (PFLASH) register reset values

Table 10-3 lists chip-specific register reset values specific to the this device. Any registers not listed have the reset values shown in their respective sections of this chapter.

Table 10-3. Chip specific flash memory controller (PFLASH) register reset values

Register	Reset Value
Platform Flash Configuration Register 1 (PFCR1)	0x601
Platform Flash Configuration Register 2 (PFCR2)	0x1
Platform Flash Configuration Register 3 (PFCR3)	0x0

NOTE

Although the PFLASH_PFCR1[RWSC] field is 5 bits wide, the MSB of this field has no effect on this chip.

10.4.3.2 Embedded Flash Memory (c55fmc) register reset values

Table 10-4 lists chip-specific register reset values specific to this device. Any registers not listed have the reset values shown in their respective sections of this chapter.

Table 10-4. Chip specific c55fmc flash memory reset values

Register	Reset Value
Module Configuration Register (C55FMC_MCR)	0000_0600h
Extended Module Configuration Register (C55FMC_MCRE)	0301_6102h [1]
Over-Program Protection 0 register (C55FMC_OPP0)	Dependent on customer settings[2]
Over-Program Protection 1 register (C55FMC_OPP1)	Dependent on customer settings[2]
Over-Program Protection 2 register (C55FMC_OPP2)	Dependent on customer settings[2]
Over-Program Protection 3 register (C55FMC_OPP3)	Dependent on customer settings[2]

1. The reset value of the C55FMC_MCRE register is device-specific. The value given applies to devices with 2 MB of flash memory.
2. The reset value of the C55FMC_OPP*n* registers, for each flash memory block mapped to a bit within each register, reflects whether the block has been marked as One Time Programmable (OTP). The C55FMC_OPP*n* registers have the same bit-to-block mapping as the C55FMC_LOCK*n* registers. A 1 indicates a block has been marked as OTP. The reset values of the C55FMC_OPP*n* are similar to the reset values of the TDM_OTPEN*n* registers in the Tamper Detection Module (TDM).

Please refer to attached flash block assignment xls for C55FMC_LOCKx memory map.

10.4.3.3 Flash memory partitioning

Refer to Flash block assignment tables attached as Excel file with the document.

10.5 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, these microcontrollers include *end-to-end ECC support*. End-to-end ECC (aka e2eECC) is structurally different than traditional "ECC at memory" functionality in that it provides for robust error detection capabilities from one endpoint of an information transfer to another with temporary information storage in any intermediate component(s). While memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors which have occurred on stored data in the memory, e2eECC

instead performs generation of error protection codes at the source of data generation, sending the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master, and performs a check of data integrity using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then may regenerate the codes for provision when a request is made to read the stored information, or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated based on more than just the data associated with a storage location in order to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the store data at the data source to generate error protection codes which cover certain types of addressing errors which may occur in the system interconnect or in the memory unit. Checking of the error protection codes may be done at the memory unit on a store to ensure that no corruption of address or data information has occurred while the request has transitioned through the device from the bus master source, as well as to ensure that within the storage memory, address decoding was performed properly (although not all address decoding errors can be detected this way), or it may simply store the received data and protection codes at the address it receives. On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location(s), and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory, or in the retrieved data, thus ensuring that the address sent for the request was not corrupted, that the addressed location was actually accessed (to the extent it is possible to ensure), and that the stored data was error free, to the extend the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in the following table.

Table 10-5. ECC comparison of data write then read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data + checkbits into the memory

Table continues on the next page...

Table 10-5. ECC comparison of data write then read sequence (continued)

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations "covered" by the ECC checks is readily apparent. Thus, the e2eECC concept provides improved fault detection capabilities in two important aspects:

1. The entire data transaction, from the initiating bus master, through the entire platform crossbar steering mechanism to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
2. The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state affecting whether the system bus transfer must be stalled (to correct a single bit error or report a non-correctable event) or is allowed to complete (for error free transfers) in a critical timing arc which often sets the upper limit of operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master, where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic, producing an improved timing arc and generally, higher operating speeds.

Errors which occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requestor. While not all possible errors can be detected this way, this approach provides a substantial

improvement versus traditional methods. Additional on-line diagnostics and/or additional hardware should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

Chapter 11
Device Configuration Format (DCF) Records

11.1 Overview

Device Configuration Format (DCF) records are used by the System Status and Configuration Module (SSCM) to configure certain registers in the device during system boot while the reset signal is asserted.

There are two kinds of DCF records:

- Factory-written DCF records: written to UTEST flash by the factory.
- User-supplied DCF records: written to UTEST flash by the user.

The following figure is a conceptual representation of the functionality of DCF records which is described in this chapter.

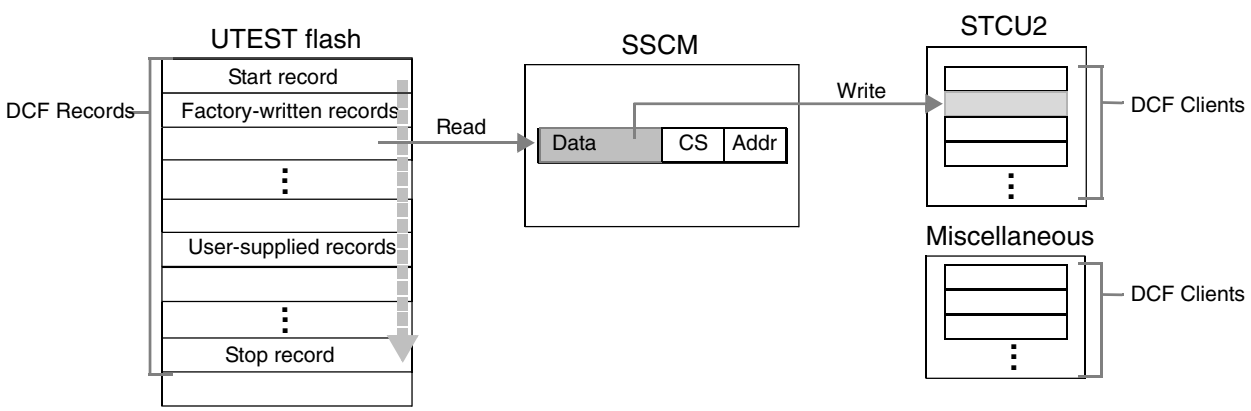


Figure 11-1. Conceptual representation of DCF records functionality

11.2 DCF records in boot sequence

For the device to be used in a user application, the system boot process must properly program the following to the respective flash memories before releasing reset.

- User application code
- Reset vectors for all CPUs
- DCF records
- Life cycle records

The following is a high-level summary of the boot sequence leading to the use of DCF records:

1. When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device.
2. After the system power supplies have reached predefined levels, the Power Management Controller signals the Reset Generation Module (RGM) to begin the boot sequence.
3. During the boot sequence, the RGM enables the System Status and Control Module (SSCM) to read the Device Configuration Format records.
4. SSCM writes the configuration information to the specified registers.
5. After registers are initialized, the SSCM passes the control of the boot sequence back to the RGM, which directs the STCU2 to start the memory and logic built in self tests (MBIST and LBIST).
6. After MBIST and LBIST are complete, RGM releases reset and device starts code execution from reset vector.

11.3 DCF records function

User-supplied DCF records define the following:

- Setup of the initial memory map
- Tests the Self-Test Control Unit (STCU2) runs during the boot sequence
- Flash memory block association with specific password groups
- Flash memory block association with specific tamper detect regions

11.4 DCF records location

Factory-written DCF records start at the first address in the UTEST flash memory area. Refer the 'UTEST Memory Map' tab in the DCF sheet attached with this document as an Excel file

User-supplied UTEST DCF records may be added at the next location in the UTEST memory map immediately following the factory-written UTEST DCF records.

11.5 DCF record structure

A DCF record is a double-word (64-bit) entry consisting of the following:

- Control word: Information to locate the corresponding DCF client internal to the device (pointer to the location of a register internal to the device)
- Data word: Data to be written to that client

DCF Records select the target DCF client by a 30-bit field in the DCF Record consisting of a 15-bit 'Chip Select' field and a 15-bit Address field. Each module that includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

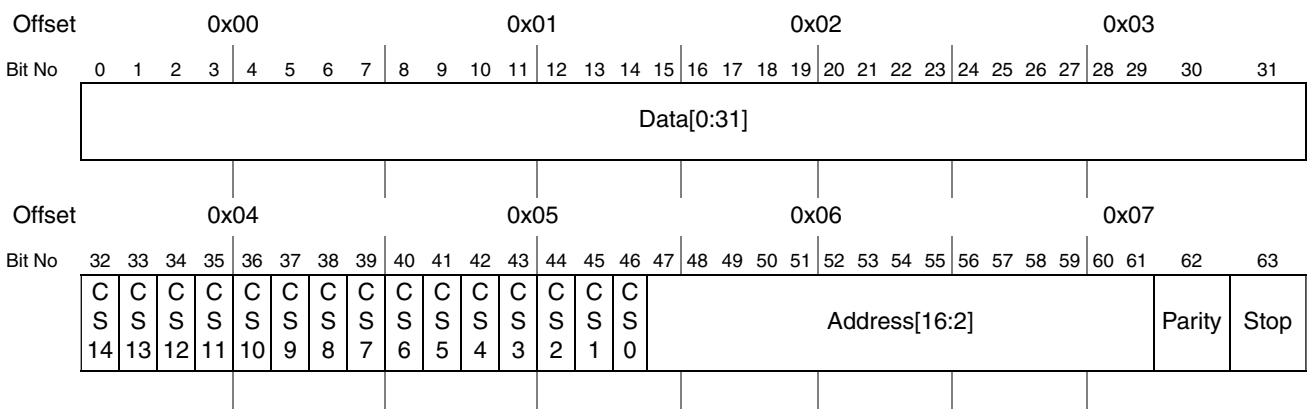


Figure 11-2. DCF Record Structure

NOTE

Attached DCF Sheet is in little endian format. Bit 31 of DCF Client in DCF Sheet is mapped to 0th Bit of Data in DCF Record.

Table 11-1. DCF Record Field Description

Field	Name	Description
0-31	Data[0:31]	32 bits of data that is to be written to the DCF Client
32-46	CS*n*	Chip Select *n* One Chip select is asserted ('1') per DCF record to select the target module for the DCF client. All other Chip Selects should be negated ('0').
47-61	Address[16:2]	Address of the DCF client within the selected module.

Table continues on the next page...

Table 11-1. DCF Record Field Description (continued)

Field	Name	Description
		Note: Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter.
62	Parity	Parity Bit for the DCF Record. Note: This bit is NOT implemented for DCF Client written from UTEST.
63	Stop	Stop bit. Indicates the end of the list of DCF Records. Stop = '0' - NOT the end of the list Stop = '1' - End of the list Note: The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list.

11.6 DCF records sequence

DCF records must appear as a contiguous series of entries programmed from the beginning of the UTEST flash memory area in the following order.

1. **Start DCF record**: The first DCF record must be a start record. This record must be found at the beginning of a DCF area in flash memory to indicate to the device that the following records must be processed.
2. **DCF records containing configuration data**: DCF records containing configuration data must immediately follow the start record with no blank records in between.
3. **Stop DCF record**: The end of the configuration records is indicated by the stop record. A record with the stop bit set is a stop record. All other bits are ignored. No DCF records following the stop record are processed.

SSCM recognizes the following record as a start record.

Table 11-2. DCF start record

0x00 (0:31)	0x04 (32:63)
0x05AA_55AF	0x0000_0000

The start record is set by the factory at the beginning of the UTEST flash memory area.

SSCM recognizes the following record as a stop record.

Table 11-3. DCF stop record

0:31	32:62	63
Ignored	Ignored	1

The stop bit is never set by the factory, since this would not allow the user to add DCF records. The flash memory location following the last factory-written UTEST DCF record is unprogrammed, with the content of 0xFFFF_FFFF. This is interpreted as the stop record, since the stop bit is set.

User-supplied DCF records may be added in a contiguous manner immediately following the factory-written DCF records. There must never be an unprogrammed record in the series of DCF records as it will be interpreted as a stop record.

The following table shows the series of DCF records when *n* data records are stored in the UTEST flash.

Table 11-4. Series of DCF records in UTEST flash memory

Record type	ADDR offset	DATA			
Start record	0x00	0x05AA_55AF			
	0x04	0x0000_0000			STOP=0
Data records	0x08	WDATA[31:0]			
	0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	0x10	WDATA[31:0]			
	0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	...	...			
Stop record	8*n*-1 + 0x0	Reserved			
	8*n*-1 + 0x4	Reserved			1
	8*n* + 0x0				
	8*n* + 0x4				

It is possible for more than one DCF Record to write to the same DCF client. In this case the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

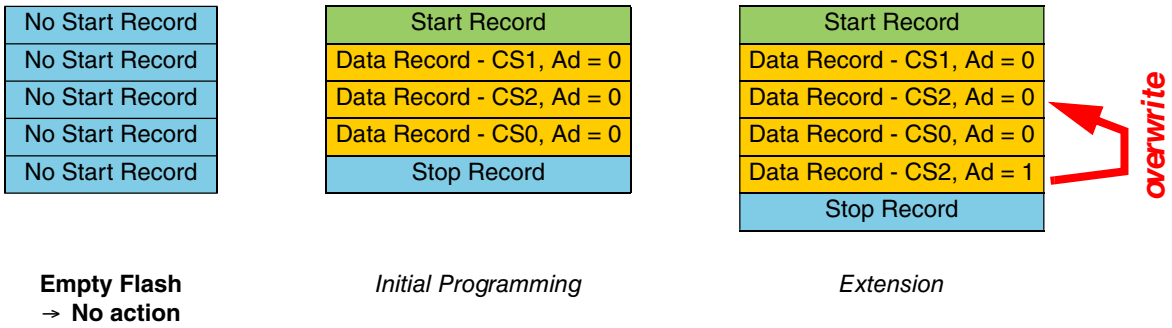


Figure 11-3. Appending DCF records

11.7 DCF clients

DCF clients are 32-bit wide hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features.

The DCF clients are described in the tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click the Excel file to open it.

Some DCF clients support a safety feature called Triple Voting. For safe DCF clients, three DCF records (called mirrors) need to be populated in Flash:

- Mirror0: Actual Data, Control Word (CS, Actual Client_Address + 0x4)
- Mirror1: Inverted Data, Control Word (CS, Client_Address + 0x8)
- Mirror2: Right Rotated (by 1 bit) Data, Control Word (CS, Client_Address + 0x10)

The number of bits to rotate depends on the DCF client parameter (Refer "# of Valid Bits" column given in the "UTEST DCF Clients" tab of the DCF heet attached with this document as an Excel file).

If WR_ONCE is one for any DCF client, it will only accept a single write.

The following DCF Bits should not be programmed multiple times

- dcf_ctl_tsens. TS0_AOUT_EN
- dcf_ctl_tsens. TS0_DOUT_EN
- dcf_ctl_tsens. TS1_AOUT_EN
- dcf_ctl_tsens. TS1_DOUT_EN

Below DCF clients are factory programmed and not recommended to be overwritten

- dcl_soc_conf_3. IO_LVD_POR

- dcl_soc_conf_3. LVD_POR
- dcl_soc_conf_3. PMC_SELFTEST_ALL_VD_MASK

Chapter 12
System Integration Unit Lite2 (SIUL2)

12.1 Introduction

12.1.1 Overview

The SIUL2 provides control over all the electrical pin controls and up to 32 ports with 16 bits of bidirectional, general-purpose input and output signals. One of the most important functions of the SIUL2 is that it enables the user to select the functions and electrical characteristics that appear on external device pins. It also controls the multiplexing of internal signals from one module to another and controls device I/O. It supports up to 32 external interrupts with trigger event configuration. The following figure is the block diagram of the SIUL2 and its interfaces to other system components.

The module provides dedicated pad control to general-purpose pads that can be configured as either inputs or outputs. The SIUL2 module provides registers that enable user software to read values from GPIO pads configured as inputs, and write values to GPIO pads configured as outputs

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

To assist software development, GPIO data registers can be accessed using various mechanisms. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16 bit ports in one access
- Read/write access to a single bit
- A 16-bit Port write with a bit mask, using single 32-bit access.

The external interrupt sources can be configured at the MCU level to be used with any chip pad. From 1 to 32 of the interrupt sources can be configured to have a digital filter to reject short glitches on the inputs. The external interrupt/DMA requests map to the REQ pins in the device packages.

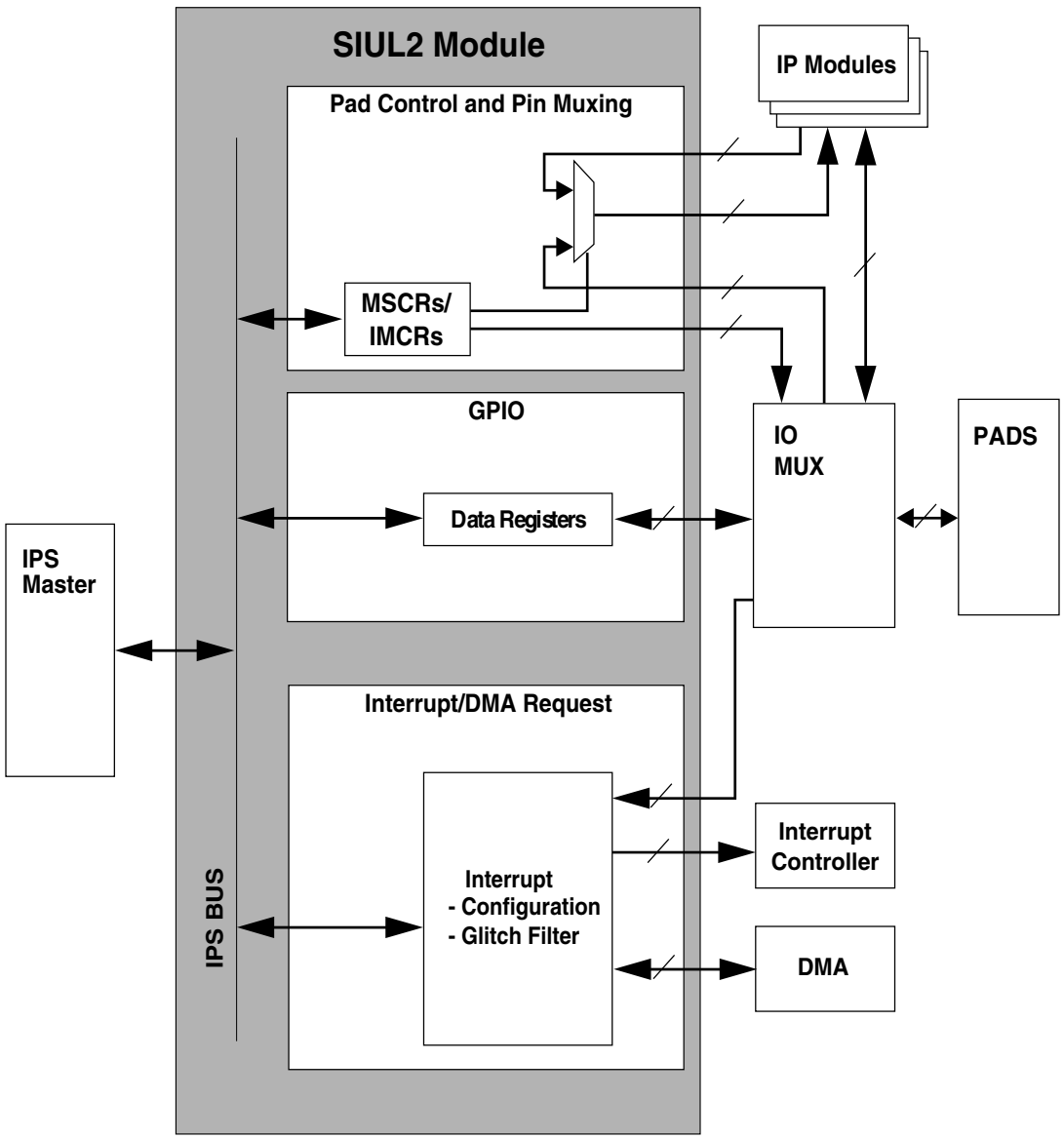


Figure 12-1. SIUL2 block diagram

12.1.2 Features

The System Integration Unit Lite2 supports these distinctive features:

- 1 to 32 GPIO ports with data control
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels

Two 16-bit registers can be read/written with one access for a 32-bit port, if needed.

External interrupt/DMA request support with:

- 1 to 4 system interrupt vectors for 1 to 32 interrupt sources with independent interrupt masks. For 32 external interrupt sources (REQ pins), four groups have eight interrupt sources each, and each of the four groups is assigned one system interrupt vector.
- 1 to 32 programmable digital glitch filters, one for each REQ pin
- 1 to 4 system DMA request channels for 1 to 32 REQ pins
- Edge detection

Additionally the SIUL2 contains the Multiplexed Signal Configuration Registers that configure the electrical parameters and settings of up to 512 functional pads. The number of these registers that is actually implemented varies by device. These registers configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control
- Hysteresis control
- Inversion control
- Internal pull control and pull selection
- Pin function assignment

- Control of analog path switches
- Safe mode behavior configuration

12.2 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL2 module in address order. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

The following table gives an overview on the SIUL2 registers implemented.

NOTE

Undocumented register spaces in the SIUL2 memory map, including addresses shown as "blank," are reserved. Reserved registers read as 0, and writes to these registers have no effect. If supported and enabled by the MCU, a transfer error is issued for attempts to access a completely reserved register space. A write to reserved register bit has no effect.

NOTE

The implementation of the MSCRn, GPDOn, GPDIn, PGPDOn and PGPDIn and MPGPDOn is not in continuous incremental order in the device. There are GPIO holes. For exact implementation, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Table 12-1. MIDR1/2 Reset values

257MAPBGA	MIDR1	MIDR2	Configuration	Performance	Temperature
FS32R274KSK2MMM	0x27402002	0x01524B42	S	K	M
FS32R274KCK2MMM	0x27402002	0x01524B43	C	K	M
FS32R274VBK2MMM	0x27402002	0x01525642	B	V	M
FS32R274VCK2MMM	0x27402002	0x01525643	C	V	M

Table continues on the next page...

Table 12-1. MIDR1/2 Reset values (continued)

257MAPBGA	MIDR1	MIDR2	Configuration	Performance	Temperature
FS32R274KSK2VMM	0x27402002	0x01524B42	S	K	V
FS32R274KCK2VMM	0x27402002	0x01524B43	C	K	V
FS32R274VBK2VMM	0x27402002	0x01525642	B	V	V
FS32R274VCK2VMM	0x27402002	0x01525643	C	V	V

Table 12-2. Configuration

Configuration	2 MB Flash	1 MB RAM	1.25 MB RAM	1.5 MB RAM	CSE
B or S	Yes	No	No	Yes	Yes
C	Yes	No	No	Yes	No

Table 12-3. Performance

Perf (MHz)	Z7	Z7	Z4	Z4
K	240	240	120	120
V	200	200	100	100

Table 12-4. Temperature values

Temperature	TA
M	-40° C to 125° C
V	-40° C to 105° C

SIUL2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	32	R	See section	12.2.1/256
8	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	32	R	See section	12.2.2/258
10	SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)	32	w1c	0000_0000h	12.2.3/259
18	SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)	32	R/W	0000_0000h	12.2.4/264
20	SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)	32	R/W	0000_0000h	12.2.5/268
28	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	32	R/W	0000_0000h	12.2.6/272
30	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	32	R/W	0000_0000h	12.2.7/275

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	32	R/W	0000_0000h	12.2.8/279
40	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	32	R/W	0000_0000h	12.2.9/282
44	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR1)	32	R/W	0000_0000h	12.2.9/282
48	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR2)	32	R/W	0000_0000h	12.2.9/282
4C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR3)	32	R/W	0000_0000h	12.2.9/282
50	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR4)	32	R/W	0000_0000h	12.2.9/282
54	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR5)	32	R/W	0000_0000h	12.2.9/282
58	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR6)	32	R/W	0000_0000h	12.2.9/282
5C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR7)	32	R/W	0000_0000h	12.2.9/282
60	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR8)	32	R/W	0000_0000h	12.2.9/282
64	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR9)	32	R/W	0000_0000h	12.2.9/282
68	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR10)	32	R/W	0000_0000h	12.2.9/282
6C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR11)	32	R/W	0000_0000h	12.2.9/282
70	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR12)	32	R/W	0000_0000h	12.2.9/282
74	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR13)	32	R/W	0000_0000h	12.2.9/282
78	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR14)	32	R/W	0000_0000h	12.2.9/282
7C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR15)	32	R/W	0000_0000h	12.2.9/282
80	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR16)	32	R/W	0000_0000h	12.2.9/282
84	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR17)	32	R/W	0000_0000h	12.2.9/282
88	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR18)	32	R/W	0000_0000h	12.2.9/282
8C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR19)	32	R/W	0000_0000h	12.2.9/282
90	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR20)	32	R/W	0000_0000h	12.2.9/282
94	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR21)	32	R/W	0000_0000h	12.2.9/282

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
98	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR22)	32	R/W	0000_0000h	12.2.9/282
9C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR23)	32	R/W	0000_0000h	12.2.9/282
A0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR24)	32	R/W	0000_0000h	12.2.9/282
A4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR25)	32	R/W	0000_0000h	12.2.9/282
A8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR26)	32	R/W	0000_0000h	12.2.9/282
AC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR27)	32	R/W	0000_0000h	12.2.9/282
B0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR28)	32	R/W	0000_0000h	12.2.9/282
B4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR29)	32	R/W	0000_0000h	12.2.9/282
B8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR30)	32	R/W	0000_0000h	12.2.9/282
BC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR31)	32	R/W	0000_0000h	12.2.9/282
C0	SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)	32	R/W	0000_0000h	12.2.10/283
240	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0)	32	R/W	See section	12.2.11/283
244	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR1)	32	R/W	See section	12.2.11/283
248	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR2)	32	R/W	See section	12.2.11/283
24C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR3)	32	R/W	See section	12.2.11/283
250	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR4)	32	R/W	See section	12.2.11/283
254	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR5)	32	R/W	See section	12.2.11/283
258	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR6)	32	R/W	See section	12.2.11/283
25C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR7)	32	R/W	See section	12.2.11/283
260	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR8)	32	R/W	See section	12.2.11/283
264	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR9)	32	R/W	See section	12.2.11/283
268	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR10)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
26C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR11)	32	R/W	See section	12.2.11/283
270	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR12)	32	R/W	See section	12.2.11/283
274	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR13)	32	R/W	See section	12.2.11/283
278	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR14)	32	R/W	See section	12.2.11/283
27C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR15)	32	R/W	See section	12.2.11/283
280	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR16)	32	R/W	See section	12.2.11/283
284	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR17)	32	R/W	See section	12.2.11/283
288	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR18)	32	R/W	See section	12.2.11/283
28C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR19)	32	R/W	See section	12.2.11/283
290	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR20)	32	R/W	See section	12.2.11/283
294	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR21)	32	R/W	See section	12.2.11/283
298	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR22)	32	R/W	See section	12.2.11/283
29C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR23)	32	R/W	See section	12.2.11/283
2A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR24)	32	R/W	See section	12.2.11/283
2A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR25)	32	R/W	See section	12.2.11/283
2A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR26)	32	R/W	See section	12.2.11/283
2AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR27)	32	R/W	See section	12.2.11/283
2B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR28)	32	R/W	See section	12.2.11/283
2B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR29)	32	R/W	See section	12.2.11/283
2B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR30)	32	R/W	See section	12.2.11/283
2BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR31)	32	R/W	See section	12.2.11/283
2C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR32)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR33)	32	R/W	See section	12.2.11/283
2C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR34)	32	R/W	See section	12.2.11/283
2CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR35)	32	R/W	See section	12.2.11/283
2D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR36)	32	R/W	See section	12.2.11/283
2D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR37)	32	R/W	See section	12.2.11/283
2D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR38)	32	R/W	See section	12.2.11/283
2DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR39)	32	R/W	See section	12.2.11/283
2E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR40)	32	R/W	See section	12.2.11/283
2E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR41)	32	R/W	See section	12.2.11/283
2E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR42)	32	R/W	See section	12.2.11/283
2EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR43)	32	R/W	See section	12.2.11/283
2F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR44)	32	R/W	See section	12.2.11/283
2F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR45)	32	R/W	See section	12.2.11/283
2F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR46)	32	R/W	See section	12.2.11/283
2FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR47)	32	R/W	See section	12.2.11/283
300	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR48)	32	R/W	See section	12.2.11/283
304	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR49)	32	R/W	See section	12.2.11/283
308	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR50)	32	R/W	See section	12.2.11/283
30C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR51)	32	R/W	See section	12.2.11/283
310	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR52)	32	R/W	See section	12.2.11/283
314	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR53)	32	R/W	See section	12.2.11/283
318	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR54)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
31C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR55)	32	R/W	See section	12.2.11/283
320	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR56)	32	R/W	See section	12.2.11/283
324	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR57)	32	R/W	See section	12.2.11/283
328	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR58)	32	R/W	See section	12.2.11/283
32C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR59)	32	R/W	See section	12.2.11/283
330	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR60)	32	R/W	See section	12.2.11/283
334	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR61)	32	R/W	See section	12.2.11/283
338	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR62)	32	R/W	See section	12.2.11/283
33C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR63)	32	R/W	See section	12.2.11/283
340	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR64)	32	R/W	See section	12.2.11/283
344	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR65)	32	R/W	See section	12.2.11/283
348	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR66)	32	R/W	See section	12.2.11/283
34C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR67)	32	R/W	See section	12.2.11/283
350	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR68)	32	R/W	See section	12.2.11/283
354	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR69)	32	R/W	See section	12.2.11/283
358	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR70)	32	R/W	See section	12.2.11/283
35C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR71)	32	R/W	See section	12.2.11/283
360	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR72)	32	R/W	See section	12.2.11/283
364	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR73)	32	R/W	See section	12.2.11/283
368	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR74)	32	R/W	See section	12.2.11/283
36C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR75)	32	R/W	See section	12.2.11/283
370	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR76)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
374	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR77)	32	R/W	See section	12.2.11/283
378	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR78)	32	R/W	See section	12.2.11/283
37C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR79)	32	R/W	See section	12.2.11/283
380	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR80)	32	R/W	See section	12.2.11/283
384	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR81)	32	R/W	See section	12.2.11/283
388	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR82)	32	R/W	See section	12.2.11/283
38C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR83)	32	R/W	See section	12.2.11/283
390	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR84)	32	R/W	See section	12.2.11/283
394	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR85)	32	R/W	See section	12.2.11/283
398	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR86)	32	R/W	See section	12.2.11/283
39C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR87)	32	R/W	See section	12.2.11/283
3A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR88)	32	R/W	See section	12.2.11/283
3A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR89)	32	R/W	See section	12.2.11/283
3A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR90)	32	R/W	See section	12.2.11/283
3AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR91)	32	R/W	See section	12.2.11/283
3B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR92)	32	R/W	See section	12.2.11/283
3B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR93)	32	R/W	See section	12.2.11/283
3B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR94)	32	R/W	See section	12.2.11/283
3BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR95)	32	R/W	See section	12.2.11/283
3C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR96)	32	R/W	See section	12.2.11/283
3C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR97)	32	R/W	See section	12.2.11/283
3C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR98)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR99)	32	R/W	See section	12.2.11/283
3D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR100)	32	R/W	See section	12.2.11/283
3D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR101)	32	R/W	See section	12.2.11/283
3D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR102)	32	R/W	See section	12.2.11/283
3DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR103)	32	R/W	See section	12.2.11/283
3E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR104)	32	R/W	See section	12.2.11/283
3E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR105)	32	R/W	See section	12.2.11/283
3E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR106)	32	R/W	See section	12.2.11/283
3EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR107)	32	R/W	See section	12.2.11/283
3F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR108)	32	R/W	See section	12.2.11/283
3F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR109)	32	R/W	See section	12.2.11/283
3F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR110)	32	R/W	See section	12.2.11/283
3FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR111)	32	R/W	See section	12.2.11/283
400	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR112)	32	R/W	See section	12.2.11/283
404	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR113)	32	R/W	See section	12.2.11/283
408	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR114)	32	R/W	See section	12.2.11/283
40C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR115)	32	R/W	See section	12.2.11/283
410	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR116)	32	R/W	See section	12.2.11/283
414	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR117)	32	R/W	See section	12.2.11/283
418	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR118)	32	R/W	See section	12.2.11/283
41C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR119)	32	R/W	See section	12.2.11/283
420	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR120)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
424	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR121)	32	R/W	See section	12.2.11/283
428	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR122)	32	R/W	See section	12.2.11/283
42C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR123)	32	R/W	See section	12.2.11/283
430	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR124)	32	R/W	See section	12.2.11/283
434	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR125)	32	R/W	See section	12.2.11/283
438	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR126)	32	R/W	See section	12.2.11/283
43C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR127)	32	R/W	See section	12.2.11/283
440	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR128)	32	R/W	See section	12.2.11/283
444	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR129)	32	R/W	See section	12.2.11/283
448	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR130)	32	R/W	See section	12.2.11/283
44C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR131)	32	R/W	See section	12.2.11/283
450	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR132)	32	R/W	See section	12.2.11/283
454	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR133)	32	R/W	See section	12.2.11/283
458	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR134)	32	R/W	See section	12.2.11/283
45C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR135)	32	R/W	See section	12.2.11/283
460	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR136)	32	R/W	See section	12.2.11/283
464	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR137)	32	R/W	See section	12.2.11/283
468	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR138)	32	R/W	See section	12.2.11/283
46C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR139)	32	R/W	See section	12.2.11/283
470	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR140)	32	R/W	See section	12.2.11/283
474	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR141)	32	R/W	See section	12.2.11/283
478	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR142)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
47C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR143)	32	R/W	See section	12.2.11/283
480	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR144)	32	R/W	See section	12.2.11/283
484	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR145)	32	R/W	See section	12.2.11/283
488	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR146)	32	R/W	See section	12.2.11/283
48C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR147)	32	R/W	See section	12.2.11/283
490	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR148)	32	R/W	See section	12.2.11/283
494	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR149)	32	R/W	See section	12.2.11/283
498	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR150)	32	R/W	See section	12.2.11/283
49C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR151)	32	R/W	See section	12.2.11/283
4A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR152)	32	R/W	See section	12.2.11/283
4A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR153)	32	R/W	See section	12.2.11/283
4A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR154)	32	R/W	See section	12.2.11/283
4AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR155)	32	R/W	See section	12.2.11/283
4B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR156)	32	R/W	See section	12.2.11/283
4B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR157)	32	R/W	See section	12.2.11/283
4B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR158)	32	R/W	See section	12.2.11/283
4BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR159)	32	R/W	See section	12.2.11/283
4C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR160)	32	R/W	See section	12.2.11/283
4C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR161)	32	R/W	See section	12.2.11/283
4C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR162)	32	R/W	See section	12.2.11/283
4CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR163)	32	R/W	See section	12.2.11/283
4D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR164)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR165)	32	R/W	See section	12.2.11/283
4D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR166)	32	R/W	See section	12.2.11/283
4DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR167)	32	R/W	See section	12.2.11/283
4E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR168)	32	R/W	See section	12.2.11/283
4E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR169)	32	R/W	See section	12.2.11/283
4E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR170)	32	R/W	See section	12.2.11/283
4EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR171)	32	R/W	See section	12.2.11/283
4F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR172)	32	R/W	See section	12.2.11/283
4F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR173)	32	R/W	See section	12.2.11/283
4F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR174)	32	R/W	See section	12.2.11/283
4FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR175)	32	R/W	See section	12.2.11/283
500	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR176)	32	R/W	See section	12.2.11/283
504	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR177)	32	R/W	See section	12.2.11/283
508	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR178)	32	R/W	See section	12.2.11/283
50C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR179)	32	R/W	See section	12.2.11/283
510	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR180)	32	R/W	See section	12.2.11/283
514	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR181)	32	R/W	See section	12.2.11/283
518	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR182)	32	R/W	See section	12.2.11/283
51C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR183)	32	R/W	See section	12.2.11/283
520	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR184)	32	R/W	See section	12.2.11/283
524	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR185)	32	R/W	See section	12.2.11/283
528	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR186)	32	R/W	See section	12.2.11/283

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
52C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR187)	32	R/W	See section	12.2.11/283
530	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR188)	32	R/W	See section	12.2.11/283
534	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR189)	32	R/W	See section	12.2.11/283
538	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR190)	32	R/W	See section	12.2.11/283
53C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR191)	32	R/W	See section	12.2.11/283
540	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR192)	32	R/W	See section	12.2.11/283
544	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR193)	32	R/W	See section	12.2.11/283
548	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR194)	32	R/W	See section	12.2.11/283
54C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR195)	32	R/W	See section	12.2.11/283
550	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR196)	32	R/W	See section	12.2.11/283
554	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR197)	32	R/W	See section	12.2.11/283
558	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR198)	32	R/W	See section	12.2.11/283
A40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR0)	32	R/W	See section	12.2.12/286
A44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR1)	32	R/W	See section	12.2.12/286
A48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR2)	32	R/W	See section	12.2.12/286
A4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR3)	32	R/W	See section	12.2.12/286
A50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR4)	32	R/W	See section	12.2.12/286
A54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR5)	32	R/W	See section	12.2.12/286
A58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR6)	32	R/W	See section	12.2.12/286
A5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR7)	32	R/W	See section	12.2.12/286
A60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR8)	32	R/W	See section	12.2.12/286
A64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR9)	32	R/W	See section	12.2.12/286

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR10)	32	R/W	See section	12.2.12/286
A6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR11)	32	R/W	See section	12.2.12/286
A70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR12)	32	R/W	See section	12.2.12/286
A74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR13)	32	R/W	See section	12.2.12/286
A78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR14)	32	R/W	See section	12.2.12/286
A7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR15)	32	R/W	See section	12.2.12/286
A80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR16)	32	R/W	See section	12.2.12/286
A84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR17)	32	R/W	See section	12.2.12/286
A88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR18)	32	R/W	See section	12.2.12/286
A8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR19)	32	R/W	See section	12.2.12/286
A90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR20)	32	R/W	See section	12.2.12/286
A94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR21)	32	R/W	See section	12.2.12/286
A98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR22)	32	R/W	See section	12.2.12/286
A9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR23)	32	R/W	See section	12.2.12/286
AA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR24)	32	R/W	See section	12.2.12/286
AA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR25)	32	R/W	See section	12.2.12/286
AA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR26)	32	R/W	See section	12.2.12/286
AAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR27)	32	R/W	See section	12.2.12/286
AB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR28)	32	R/W	See section	12.2.12/286
AB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR29)	32	R/W	See section	12.2.12/286
AB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR30)	32	R/W	See section	12.2.12/286
ABC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR31)	32	R/W	See section	12.2.12/286

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR32)	32	R/W	See section	12.2.12/286
AC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR33)	32	R/W	See section	12.2.12/286
AC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR34)	32	R/W	See section	12.2.12/286
ACC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR35)	32	R/W	See section	12.2.12/286
AD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR36)	32	R/W	See section	12.2.12/286
AD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR37)	32	R/W	See section	12.2.12/286
AD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR38)	32	R/W	See section	12.2.12/286
ADC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR39)	32	R/W	See section	12.2.12/286
AE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR40)	32	R/W	See section	12.2.12/286
AE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR41)	32	R/W	See section	12.2.12/286
AE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR42)	32	R/W	See section	12.2.12/286
AEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR43)	32	R/W	See section	12.2.12/286
AF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR44)	32	R/W	See section	12.2.12/286
AF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR45)	32	R/W	See section	12.2.12/286
AF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR46)	32	R/W	See section	12.2.12/286
AFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR47)	32	R/W	See section	12.2.12/286
B00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR48)	32	R/W	See section	12.2.12/286
B04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR49)	32	R/W	See section	12.2.12/286
B08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR50)	32	R/W	See section	12.2.12/286
B0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR51)	32	R/W	See section	12.2.12/286
B10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR52)	32	R/W	See section	12.2.12/286
B14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR53)	32	R/W	See section	12.2.12/286

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR54)	32	R/W	See section	12.2.12/286
B1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR55)	32	R/W	See section	12.2.12/286
B20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR56)	32	R/W	See section	12.2.12/286
B24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR57)	32	R/W	See section	12.2.12/286
B28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR58)	32	R/W	See section	12.2.12/286
B2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR59)	32	R/W	See section	12.2.12/286
B30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR60)	32	R/W	See section	12.2.12/286
B34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR61)	32	R/W	See section	12.2.12/286
B38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR62)	32	R/W	See section	12.2.12/286
B3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR63)	32	R/W	See section	12.2.12/286
B40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR64)	32	R/W	See section	12.2.12/286
B44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR65)	32	R/W	See section	12.2.12/286
B48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR66)	32	R/W	See section	12.2.12/286
B4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR67)	32	R/W	See section	12.2.12/286
B50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR68)	32	R/W	See section	12.2.12/286
B54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR69)	32	R/W	See section	12.2.12/286
B58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR70)	32	R/W	See section	12.2.12/286
B5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR71)	32	R/W	See section	12.2.12/286
B60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR72)	32	R/W	See section	12.2.12/286
B64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR73)	32	R/W	See section	12.2.12/286
B68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR74)	32	R/W	See section	12.2.12/286
B6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR75)	32	R/W	See section	12.2.12/286

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR76)	32	R/W	See section	12.2.12/286
B74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR77)	32	R/W	See section	12.2.12/286
B78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR78)	32	R/W	See section	12.2.12/286
B7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR79)	32	R/W	See section	12.2.12/286
B80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR80)	32	R/W	See section	12.2.12/286
B84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR81)	32	R/W	See section	12.2.12/286
B88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR82)	32	R/W	See section	12.2.12/286
B8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR83)	32	R/W	See section	12.2.12/286
B90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR84)	32	R/W	See section	12.2.12/286
B94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR85)	32	R/W	See section	12.2.12/286
B98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR86)	32	R/W	See section	12.2.12/286
B9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR87)	32	R/W	See section	12.2.12/286
BA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR88)	32	R/W	See section	12.2.12/286
BA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR89)	32	R/W	See section	12.2.12/286
BA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR90)	32	R/W	See section	12.2.12/286
BAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR91)	32	R/W	See section	12.2.12/286
BB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR92)	32	R/W	See section	12.2.12/286
BB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR93)	32	R/W	See section	12.2.12/286
1300	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO0_3)	32	R/W	0000_0000h	12.2.13/288
1304	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO4_7)	32	R/W	0000_0000h	12.2.13/288
1308	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO8_11)	32	R/W	0000_0000h	12.2.13/288
130C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO12_15)	32	R/W	0000_0000h	12.2.13/288
1310	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO16_19)	32	R/W	0000_0000h	12.2.13/288
1314	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO20_23)	32	R/W	0000_0000h	12.2.13/288

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1318	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO24_27)	32	R/W	0000_0000h	12.2.13/288
131C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO28_31)	32	R/W	0000_0000h	12.2.13/288
1320	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO32_35)	32	R/W	0000_0000h	12.2.13/288
1324	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO36_39)	32	R/W	0000_0000h	12.2.13/288
1328	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO40_43)	32	R/W	0000_0000h	12.2.13/288
132C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO44_47)	32	R/W	0000_0000h	12.2.13/288
1330	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO48_51)	32	R/W	0000_0000h	12.2.13/288
1334	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO52_55)	32	R/W	0000_0000h	12.2.13/288
1338	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO56_59)	32	R/W	0000_0000h	12.2.13/288
133C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO60_63)	32	R/W	0000_0000h	12.2.13/288
1340	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO64_67)	32	R/W	0000_0000h	12.2.13/288
1344	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO68_71)	32	R/W	0000_0000h	12.2.13/288
1348	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO72_75)	32	R/W	0000_0000h	12.2.13/288
134C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO76_79)	32	R/W	0000_0000h	12.2.13/288
1350	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO80_83)	32	R/W	0000_0000h	12.2.13/288
1354	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO84_87)	32	R/W	0000_0000h	12.2.13/288
1358	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO88_91)	32	R/W	0000_0000h	12.2.13/288
135C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO92_95)	32	R/W	0000_0000h	12.2.13/288
1360	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO96_99)	32	R/W	0000_0000h	12.2.13/288
1364	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO100_103)	32	R/W	0000_0000h	12.2.13/288
1368	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO104_107)	32	R/W	0000_0000h	12.2.13/288
136C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO108_111)	32	R/W	0000_0000h	12.2.13/288

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1370	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO112_115)	32	R/W	0000_0000h	12.2.13/288
1374	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO116_119)	32	R/W	0000_0000h	12.2.13/288
1378	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO120_123)	32	R/W	0000_0000h	12.2.13/288
137C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO124_127)	32	R/W	0000_0000h	12.2.13/288
1380	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO128_131)	32	R/W	0000_0000h	12.2.13/288
1384	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO132_135)	32	R/W	0000_0000h	12.2.13/288
1388	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO136_139)	32	R/W	0000_0000h	12.2.13/288
138C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO140_143)	32	R/W	0000_0000h	12.2.13/288
1390	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO144_147)	32	R/W	0000_0000h	12.2.13/288
1394	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO148_151)	32	R/W	0000_0000h	12.2.13/288
1398	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO152_155)	32	R/W	0000_0000h	12.2.13/288
139C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO156_159)	32	R/W	0000_0000h	12.2.13/288
13A0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO160_163)	32	R/W	0000_0000h	12.2.13/288
13A4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO164_167)	32	R/W	0000_0000h	12.2.13/288
13A8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO168_171)	32	R/W	0000_0000h	12.2.13/288
13AC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO172_175)	32	R/W	0000_0000h	12.2.13/288
13B0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO176_179)	32	R/W	0000_0000h	12.2.13/288
13B4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO180_183)	32	R/W	0000_0000h	12.2.13/288
13B8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO184_187)	32	R/W	0000_0000h	12.2.13/288
13BC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO188_191)	32	R/W	0000_0000h	12.2.13/288
13C0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO192_195)	32	R/W	0000_0000h	12.2.13/288
13C4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO)	32	R/W	0000_0000h	12.2.13/288
1500	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI0_3)	32	R	0000_0000h	12.2.14/289
1504	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI4_7)	32	R	0000_0000h	12.2.14/289

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1508	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI8_11)	32	R	0000_0000h	12.2.14/289
150C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI12_15)	32	R	0000_0000h	12.2.14/289
1510	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI16_19)	32	R	0000_0000h	12.2.14/289
1514	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI20_23)	32	R	0000_0000h	12.2.14/289
1518	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI24_27)	32	R	0000_0000h	12.2.14/289
151C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI28_31)	32	R	0000_0000h	12.2.14/289
1520	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI32_35)	32	R	0000_0000h	12.2.14/289
1524	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI36_39)	32	R	0000_0000h	12.2.14/289
1528	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI40_43)	32	R	0000_0000h	12.2.14/289
152C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI44_47)	32	R	0000_0000h	12.2.14/289
1530	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI48_51)	32	R	0000_0000h	12.2.14/289
1534	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI52_55)	32	R	0000_0000h	12.2.14/289
1538	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI56_59)	32	R	0000_0000h	12.2.14/289
153C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI60_63)	32	R	0000_0000h	12.2.14/289
1540	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI64_67)	32	R	0000_0000h	12.2.14/289
1544	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI68_71)	32	R	0000_0000h	12.2.14/289
1548	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI72_75)	32	R	0000_0000h	12.2.14/289
154C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI76_79)	32	R	0000_0000h	12.2.14/289
1550	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI80_83)	32	R	0000_0000h	12.2.14/289
1554	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI84_87)	32	R	0000_0000h	12.2.14/289
1558	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI88_91)	32	R	0000_0000h	12.2.14/289
155C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI92_95)	32	R	0000_0000h	12.2.14/289
1560	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI96_99)	32	R	0000_0000h	12.2.14/289
1564	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI100_103)	32	R	0000_0000h	12.2.14/289
1568	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI104_107)	32	R	0000_0000h	12.2.14/289
156C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI108_111)	32	R	0000_0000h	12.2.14/289
1570	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI112_115)	32	R	0000_0000h	12.2.14/289
1574	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI116_119)	32	R	0000_0000h	12.2.14/289
1578	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI120_123)	32	R	0000_0000h	12.2.14/289
157C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI124_127)	32	R	0000_0000h	12.2.14/289
1580	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI128_131)	32	R	0000_0000h	12.2.14/289
1584	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI132_135)	32	R	0000_0000h	12.2.14/289

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1588	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI136_139)	32	R	0000_0000h	12.2.14/289
158C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI140_143)	32	R	0000_0000h	12.2.14/289
1590	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI144_147)	32	R	0000_0000h	12.2.14/289
1594	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI148_151)	32	R	0000_0000h	12.2.14/289
1598	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI152_155)	32	R	0000_0000h	12.2.14/289
159C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI156_159)	32	R	0000_0000h	12.2.14/289
15A0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI160_163)	32	R	0000_0000h	12.2.14/289
15A4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI164_167)	32	R	0000_0000h	12.2.14/289
15A8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI168_171)	32	R	0000_0000h	12.2.14/289
15AC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI172_175)	32	R	0000_0000h	12.2.14/289
15B0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI176_179)	32	R	0000_0000h	12.2.14/289
15B4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI180_183)	32	R	0000_0000h	12.2.14/289
15B8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI184_187)	32	R	0000_0000h	12.2.14/289
15BC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI188_191)	32	R	0000_0000h	12.2.14/289
15C0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI192_195)	32	R	0000_0000h	12.2.14/289
15C4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI)	32	R	0000_0000h	12.2.14/289
1700	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)	16	R/W	0000h	12.2.15/291
1702	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO1)	16	R/W	0000h	12.2.15/291
1704	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO2)	16	R/W	0000h	12.2.15/291
1706	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO3)	16	R/W	0000h	12.2.15/291
1708	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO4)	16	R/W	0000h	12.2.15/291
170A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO5)	16	R/W	0000h	12.2.15/291
170C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO6)	16	R/W	0000h	12.2.15/291

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
170E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO7)	16	R/W	0000h	12.2.15/291
1710	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO8)	16	R/W	0000h	12.2.15/291
1712	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO9)	16	R/W	0000h	12.2.15/291
1714	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO10)	16	R/W	0000h	12.2.15/291
1716	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO11)	16	R/W	0000h	12.2.15/291
1718	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO12)	16	R/W	0000h	12.2.15/291
1740	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)	16	R	0000h	12.2.16/292
1742	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI1)	16	R	0000h	12.2.16/292
1744	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI2)	16	R	0000h	12.2.16/292
1746	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI3)	16	R	0000h	12.2.16/292
1748	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI4)	16	R	0000h	12.2.16/292
174A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI5)	16	R	0000h	12.2.16/292
174C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI6)	16	R	0000h	12.2.16/292
174E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI7)	16	R	0000h	12.2.16/292
1750	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI8)	16	R	0000h	12.2.16/292
1752	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI9)	16	R	0000h	12.2.16/292
1754	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI10)	16	R	0000h	12.2.16/292
1756	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI11)	16	R	0000h	12.2.16/292
1758	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI12)	16	R	0000h	12.2.16/292
1780	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)	32	W	0000_0000h	12.2.17/293
1784	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO1)	32	W	0000_0000h	12.2.17/293
1788	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO2)	32	W	0000_0000h	12.2.17/293
178C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO3)	32	W	0000_0000h	12.2.17/293
1790	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO4)	32	W	0000_0000h	12.2.17/293
1794	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO5)	32	W	0000_0000h	12.2.17/293
1798	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO6)	32	W	0000_0000h	12.2.17/293
179C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO7)	32	W	0000_0000h	12.2.17/293

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17A0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO8)	32	W	0000_0000h	12.2.17/293
17A4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO9)	32	W	0000_0000h	12.2.17/293
17A8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO10)	32	W	0000_0000h	12.2.17/293
17AC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO11)	32	W	0000_0000h	12.2.17/293
17B0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO12)	32	W	0000_0000h	12.2.17/293

12.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register holds identification information about the device.

NOTE

This register is accessible only in 32-bit mode.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PART_FAMILY												Reserved			
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		PKG		Reserved				MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0	*	*	0	0	0	0	*	*	*	*	*	*	*	*

* Notes:

- MINOR_MASK field: Depends on device version. For reset value, see Table 12-1
- MAJOR_MASK field: Depends on device version. For reset value, see Table 12-1
- PKG field: Depends on device version. For reset value, see Table 12-1
- PART_FAMILY field: Depends on device version. For reset value, see Table 12-1

SIUL2_MIDR1 field descriptions

Field	Description
0–11 PART_FAMILY	MCU Part Number
12–17 Reserved	This field is reserved.

Table continues on the next page...

SIUL2_MIDR1 field descriptions (continued)

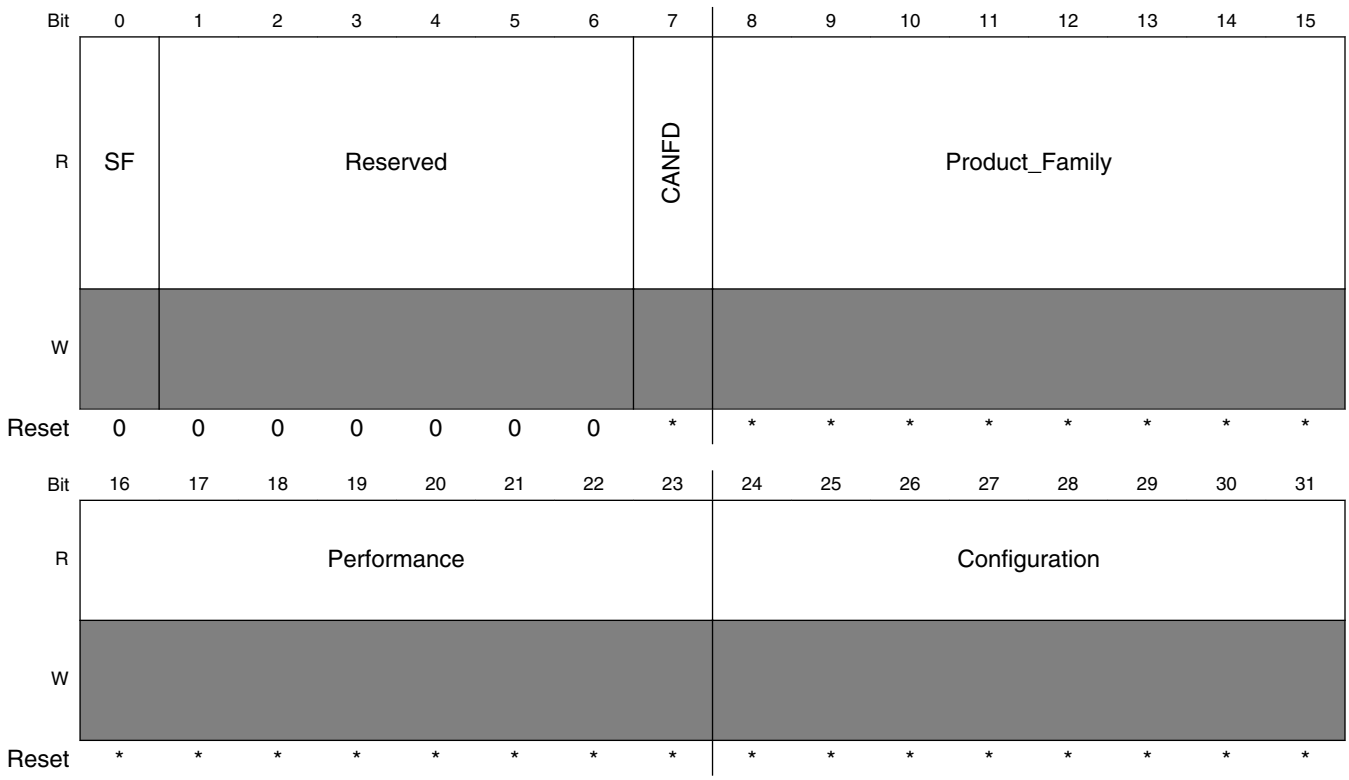
Field	Description
18–19 PKG	Package setting Can by read by software to determine the package type that is used for the particular device: 10: 257-pin BGA
20–23 Reserved	This field is reserved.
24–27 MAJOR_MASK	Major Mask Revision
28–31 MINOR_MASK	Minor Mask Revision

12.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

NOTE

This register is accessible only in 32-bit mode.

Address: 0h base + 8h offset = 8h



* Notes:
- Configuration field: Depends on device version. For reset value, see Table 12-1
- Performance field: Depends on device version. For reset value, see Table 12-1
- Product_Family field: Depends on device version. For reset value, see Table 12-1
- CANFD field: Depends on device version. For reset value, see Table 12-1

SIUL2_MIDR2 field descriptions

Field	Description
0 SF	Manufacturer 0 NXP Semiconductor, Inc 1 Reserved
1–6 Reserved	This field is reserved.
7 CANFD	CANFD

Table continues on the next page...

SIUL2_MIDR2 field descriptions (continued)

Field	Description
	0 CANFD is not supported 1 CANFD is supported
8–15 Product_Family	Product Family Number
16–23 Performance	Performance of device, see Table 12-3
24–31 Configuration	Configuration of device, see Table 12-2

12.2.3 SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set regardless of the state of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) - Standard. The IRQ Flag bit remains set until cleared by software or through the servicing of a DMA request. The IRQ Flag bits are cleared by writing a '1' to the bits. A write of '0' has no effect.

This register hold the interrupt flags.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIF31	EIF30	EIF29	EIF28	EIF27	EIF26	EIF25	EIF24	EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DISR0 field descriptions

Field	Description
0 EIF31	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
1 EIF30	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
2 EIF29	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
3 EIF28	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
4 EIF27	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
5 EIF26	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
6 EIF25	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
7 EIF24	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
8 EIF23	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
9 EIF22	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
10 EIF21	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
11 EIF20	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
12 EIF19	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
13 EIF18	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
14 EIF17	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
15 EIF16	External Interrupt Status Flag x

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
16 EIF15	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
17 EIF14	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
18 EIF13	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
19 EIF12	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
20 EIF11	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
21 EIF10	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
22 EIF9	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
23 EIF8	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
24 EIF7	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
25 EIF6	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
26 EIF5	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
27 EIF4	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
28 EIF3	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
29 EIF2	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
30 EIF1	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
31 EIF0	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

12.2.4 SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). This register is used to enable the interrupt messaging to the interrupt controller.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EIRE31	EIRE30	EIRE29	EIRE28	EIRE27	EIRE26	EIRE25	EIRE24	EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DIRER0 field descriptions

Field	Description
0 EIRE31	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
1 EIRE30	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
2 EIRE29	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
3 EIRE28	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
4 EIRE27	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
5 EIRE26	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
6 EIRE25	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
7 EIRE24	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
8 EIRE23	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes interrupt request depending on DIRSR register
9 EIRE22	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes interrupt request depending on DIRSR register
10 EIRE21	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
11 EIRE20	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
12 EIRE19	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
13 EIRE18	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
14 EIRE17	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
15 EIRE16	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
16 EIRE15	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
17 EIRE14	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
18 EIRE13	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes interrupt request depending on DIRSR register
19 EIRE12	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
20 EIRE11	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes interrupt request depending on DIRSR register
21 EIRE10	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
22 EIRE9	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
23 EIRE8	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit an interrupt request depending on DIRSR register
24 EIRE7	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
25 EIRE6	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
26 EIRE5	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
27 EIRE4	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on DIRSR register
28 EIRE3	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
29 EIRE2	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
30 EIRE1	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
31 EIRE0	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register

12.2.5 SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or a interrupt request is asserted. The number of bits implemented is specified by the parameter that defines the number of DMA request outputs.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DIRSR31	DIRSR30	DIRSR29	DIRSR28	DIRSR27	DIRSR26	DIRSR25	DIRSR24	DIRSR23	DIRSR22	DIRSR21	DIRSR20	DIRSR19	DIRSR18	DIRSR17	DIRSR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DIRSR15	DIRSR14	DIRSR13	DIRSR12	DIRSR11	DIRSR10	DIRSR9	DIRSR8	DIRSR7	DIRSR6	DIRSR5	DIRSR4	DIRSR3	DIRSR2	DIRSR1	DIRSR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DIRSR0 field descriptions

Field	Description
0 DIRSR31	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
1 DIRSR30	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
2 DIRSR29	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
3 DIRSR28	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Reserved
4 DIRSR27	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
5 DIRSR26	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
6 DIRSR25	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
7 DIRSR24	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
8 DIRSR23	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
9 DIRSR22	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
10 DIRSR21	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
11 DIRSR20	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
12 DIRSR19	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
13 DIRSR18	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Reserved
14 DIRSR17	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
15 DIRSR16	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
16 DIRSR15	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
17 DIRSR14	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
18 DIRSR13	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
19 DIRSR12	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
20 DIRSR11	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
21 DIRSR10	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
22 DIRSR9	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
23 DIRSR8	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Reserved
24 DIRSR7	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
25 DIRSR6	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
26 DIRSR5	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
27 DIRSR4	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
28 DIRSR3	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
29 DIRSR2	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
30 DIRSR1	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
31 DIRSR0	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

12.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IREE31	IREE30	IREE29	IREE28	IREE27	IREE26	IREE25	IREE24	IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IREER0 field descriptions

Field	Description
0 IREE31	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
1 IREE30	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
2 IREE29	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
3 IREE28	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
4 IREE27	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
5 IREE26	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 IREE25	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
7 IREE24	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
8 IREE23	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
9 IREE22	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
10 IREE21	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
11 IREE20	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
12 IREE19	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
13 IREE18	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 IREE17	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
15 IREE16	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
16 IREE15	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
17 IREE14	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
18 IREE13	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
19 IREE12	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
20 IREE11	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
21 IREE10	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 IREE9	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
23 IREE8	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24 IREE7	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
25 IREE6	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
26 IREE5	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
27 IREE4	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28 IREE3	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
29 IREE2	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 IREE1	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
31 IREE0	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

12.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

NOTE

If both IREE bit and IFEE bit is cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IFEE31	IFEE30	IFEE29	IFEE28	IFEE27	IFEE26	IFEE25	IFEE24	IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFEER0 field descriptions

Field	Description
0 IFEE31	Enable falling-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
1 IFEE30	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
2 IFEE29	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
3 IFEE28	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
4 IFEE27	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
5 IFEE26	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
6 IFEE25	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 IFEE24	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
8 IFEE23	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
9 IFEE22	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
10 IFEE21	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
11 IFEE20	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
12 IFEE19	Enable falling-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
13 IFEE18	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
14 IFEE17	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 IFEE16	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
16 IFEE15	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
17 IFEE14	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
18 IFEE13	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
19 IFEE12	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
20 IFEE11	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
21 IFEE10	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
22 IFEE9	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 IFEE8	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24 IFEE7	Enable falling-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
25 IFEE6	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
26 IFEE5	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
27 IFEE4	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28 IFEE3	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
29 IFEE2	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
30 IFEE1	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 IFEE0	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

12.2.8 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs. The number of interrupts supporting this feature is MCU dependent and can be configured between 1 and 32.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IFE31	IFE30	IFE29	IFE28	IFE27	IFE26	IFE25	IFE24	IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFER0 field descriptions

Field	Description
0 IFE31	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
1 IFE30	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
2 IFE29	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
3 IFE28	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
4 IFE27	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
5 IFE26	Enable digital glitch filter on the interrupt pad input.

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
	0 Filter is disabled 1 Filter is enabled
6 IFE25	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
7 IFE24	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
8 IFE23	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
9 IFE22	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
10 IFE21	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
11 IFE20	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
12 IFE19	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
13 IFE18	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
14 IFE17	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
15 IFE16	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
16 IFE15	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
17 IFE14	Enable digital glitch filter on the interrupt pad input.

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
	0 Filter is disabled 1 Filter is enabled
18 IFE13	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
19 IFE12	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
20 IFE11	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
22 IFE9	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
23 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
24 IFE7	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
25 IFE6	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28 IFE3	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
29 IFE2	Enable digital glitch filter on the interrupt pad input.

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
	0 Filter is disabled 1 Filter is enabled
30 IFE1	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
31 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

12.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

Address: 0h base + 40h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												MAXCNT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFMCR*n* field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MAXCNT	Maximum Interrupt Filter Counter setting MAXCNT can be 0d to 15d. • For a MAXCNT value of 0d, 1d, or 2d, the filter behaves as an ALL PASS filter. • For a MAXCNT value of 3d to 15d, the filter period is TCK*MAXCNT+n*TCK, where n is 0, 1, 2, 3, or 4. TCK is the prescaled filter clock period, which is the IRC clock prescaled to the IFCP value, which is specified in SIUL2_IFCPR. TIRC is the basic filter clock period: 62.5 ns (f = 16 MHz). **NOTE:** The filter delay is 2 TCK clock cycles more than the filter period. In general, TFILTER and TDELAY are not integer multiples of TCK because DIN is asynchronous whereas DOUT is synchronous with respect to the CK clock. The factor n accounts for this uncertainty in filter period calculation.

12.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filter. Prescaler is applied to the input clock to the SIUL2, which is the system AIPS clock counters in the SIUL2.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												IFCP			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFCPR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

12.2.11 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)

NOTE

This register is accessible only in 32-bit mode.

Selects which source signal is connected to the register's associated destination, which is a chip pin that is or can be configured as an output.

For the associated chip-pin destination, the register also specifies the pin's electrical properties.

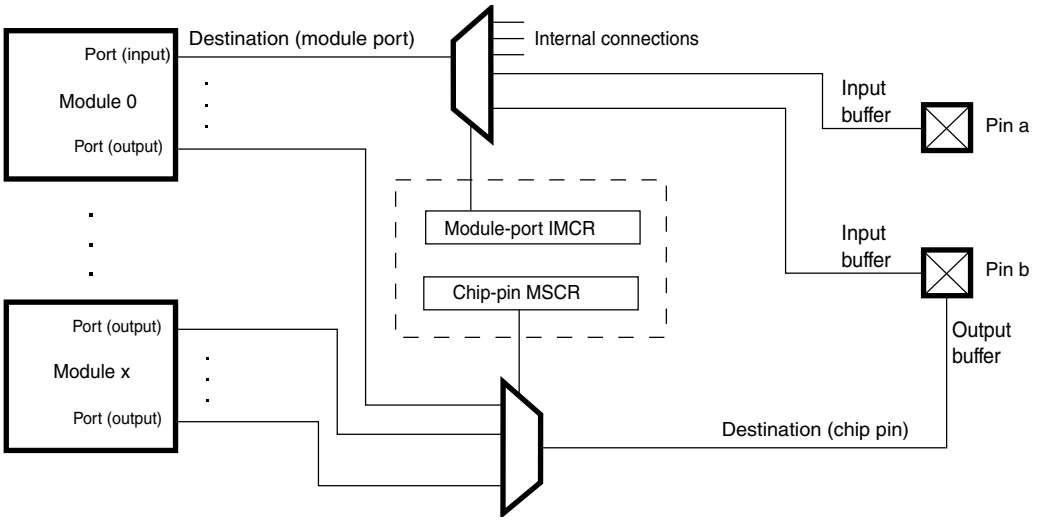


Figure 12-2. MSCR pin connection

The fields in an MSCR vary depending on its associated destination (chip pin).

For chip-pin MSCR assignments, pin types, APC support, and SSS values, see the pin-description table for your chip.

For the implemented register bits of MSCR for each pad, see the device I/O Signal Description and Input Multiplexing Tables. The fields in an MSCR vary depending on its associated destination (chip pin).

For details on the implemented chip-pin MSCR assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + 240h offset + (4d × i), where i=0d to 198d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		SRC		0		OBE	ODE	SMC	APC	0		IBE	HYS	PUS	PUE
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the Signal Description details for the reset values and the availability of pad setting bits in MSCRs.

SIUL2_MSCR*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SRC	Slew Rate Control Used only when the associated destination is a chip pin. 00 Half drive strength with slew rate control 01 Full drive strength with slew rate control 10 Half drive strength without slew rate control 11 Full drive strength without slew rate control
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 OBE	GPIO Output Buffer Enable Applies only to digital pins. Otherwise this bit is reserved. 0 Output driver disabled 1 Output driver enabled
7 ODE	Open Drain Enable **NOTE** To enable open drain both OBE and ODE bits need to be set. 0 Open drain function disabled 1 Open drain function enabled
8 SMC	Safe Mode Control Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control Used only when the associated destination is a chip pin that supports analog I/O. Enables the pin's analog-input-path switch. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 IBE	Input Buffer Enable Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. IBE bit shall not be set when APC bit is set. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin.

Table continues on the next page...

SIUL2_MSCR*n* field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
14 PUS	Pull Select Determines whether the pull function is a pullup or pulldown when the pull function is enabled by the PUE field. Used only when the associated destination is a chip pin. 0 Pulldown 1 Pullup
15 PUE	Pull Enable Enables the pull function. Used only when the associated destination is a chip pin. 0 Disabled 1 Enabled
16 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin). 0 Don't invert 1 Invert
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin). This field has a variable width depending on the number of alternate functions available for that pin/port. For a chip pin, the source signals are outputs from module ports.

12.2.12 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR*n*)

NOTE

This register is accessible only in 32-bit mode.

Selects which source signal is connected to the register's associated destination, which is an internal module port that is or can be configured as an input.

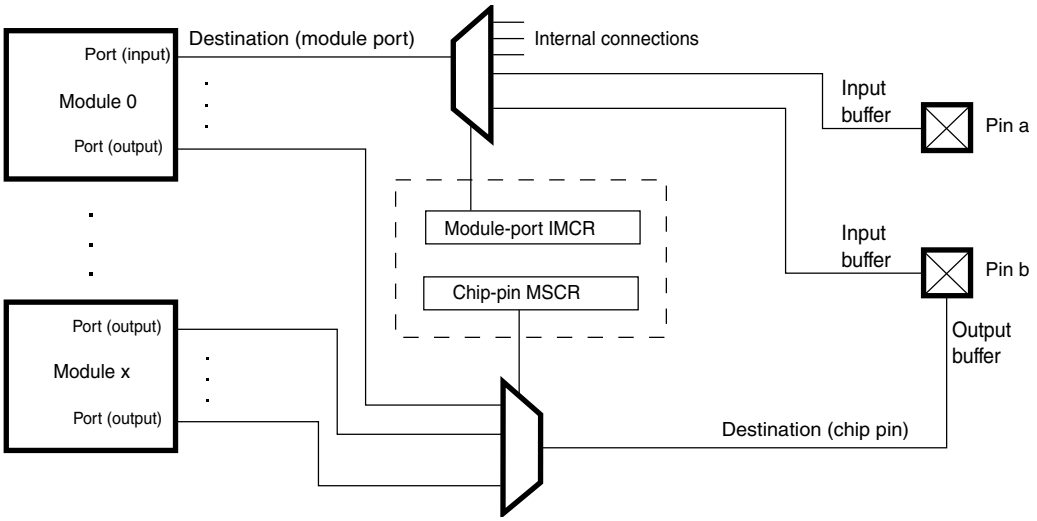


Figure 12-3. IMCR module-port connection

The fields in an IMCR vary depending on its associated destination (module port).

For IMCR assignments and SSS values, see the pin-description table for your chip.

For the implemented register bits of IMCR for each pad, see the device I/O Signal Description and Input Multiplexing Tables. The fields in an MSCR vary depending on its associated destination (chip pin).

For details on the implemented chip-pin MSCR assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + A40h offset + (4d × i), where i=0d to 93d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INV	0														
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the Signal Description details for the reset values of the IMCRs.

SIUL2_IMCR*n* field descriptions

Field	Description
0 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (module port).

Table continues on the next page...

SIUL2_IMCR*n* field descriptions (continued)

Field	Description
	0 Don't invert 1 Invert
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (module port). This field has a variable width depending on the number of alternate functions available for that pin/port. For a module port, the source signals are either outputs from module ports or inputs from chip pins.

12.2.13 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access.

For details on the implemented chip-pin GPDO assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + 1300h offset + (4d × i), where i=0d to 49d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							PDO_4n	0							PDO_4n1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							PDO_4n2	0							PDO_4n3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPDO*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIUL2_GPDO*n* field descriptions (continued)

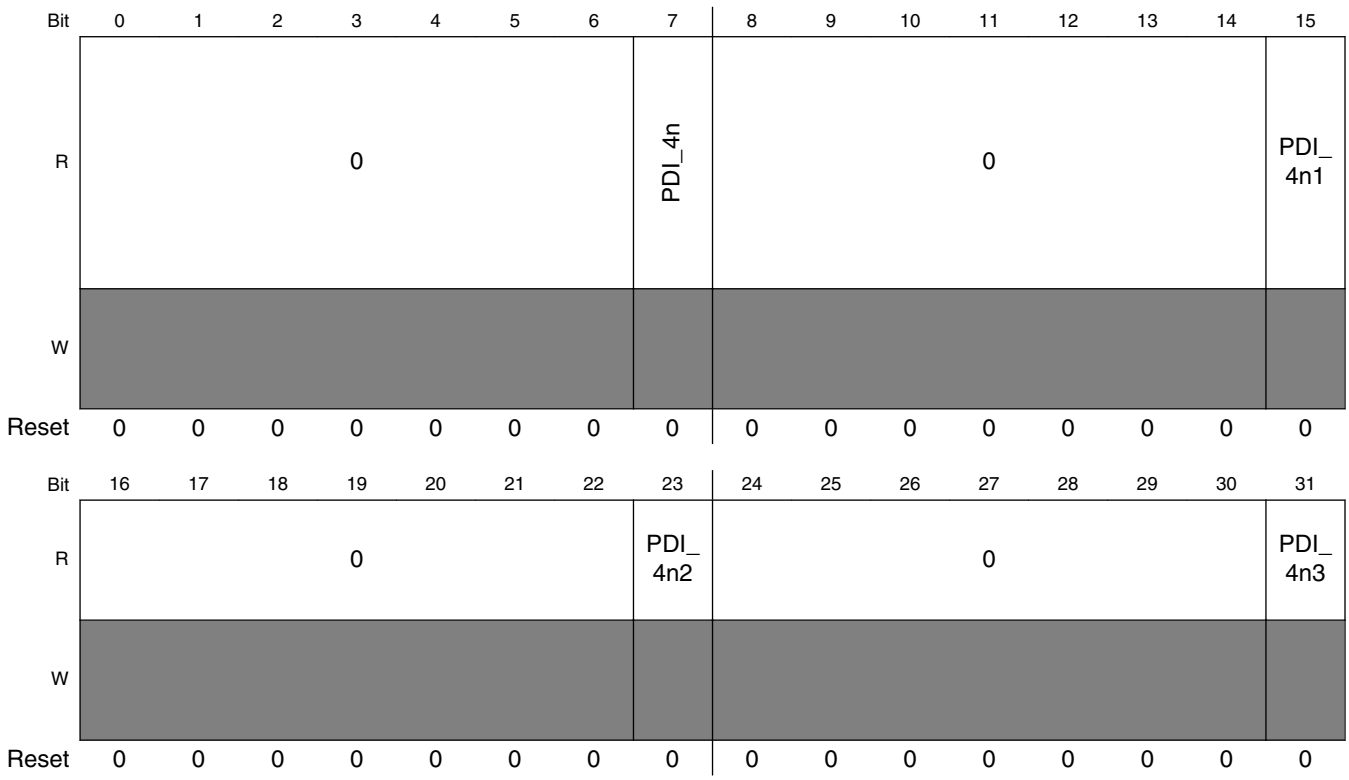
Field	Description
7 PDO_4n	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n represents PDO[4n], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n represents PDO12. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDO_4n1	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n1 represents PDO[4n+1], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n1 represents PDO13. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDO_4n2	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n2 represents PDO[4n+2], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n2 represents PDO14. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDO_4n3	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n3 represents PDO[4n+3], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n3 represents PDO15. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

12.2.14 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access.

For details on the implemented chip-pin GPDI assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + 1500h offset + (4d × i), where i=0d to 49d



SIUL2_GPDI*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI_4n	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n represents PDI[4n], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n represents PDI12. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDI_4n1	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n1 represents PDI[4n+1], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n1 represents PDI13. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDI_4n2	Pad Data In

Table continues on the next page...

SIUL2_GPDI*n* field descriptions (continued)

Field	Description
	This bit stores the value of the external GPIO pad associated with this register. PDI_4n2 represents PDI[4n+2], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n2 represents PDI14. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDI_4n3	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n3 represents PDI[4n+3], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n3 represents PDI15. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

12.2.15 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write.

The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

For details on the implemented chip-pin PGPDO assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + 1700h offset + (2d × i), where i=0d to 12d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	PPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDO*n* field descriptions

Field	Description
0–15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO).The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

12.2.16 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read.

For details on the implemented chip-pin PGPDI assignments, see the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Address: 0h base + 1740h offset + (2d × i), where i=0d to 12d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	PPDI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDI*n* field descriptions

Field	Description
0–15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

12.2.17 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32 bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0.

Address: 0h base + 1780h offset + (4d × i), where i=0d to 12d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MPGPDO*n* field descriptions

Field	Description
0–15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0b The associated bit value in the MPPDO[x] field is ignored 1b The associated bit value in the MPPDO[x] field is written
16–31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO).The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

12.3 Functional description

12.3.1 General

This section provides a complete functional description of the System Integration Unit Lite2.

12.3.2 Pad Control

The SIUL2 is capable of controlling the electrical characteristic of up to 512 pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features will be supported on all devices or on all pads as this is dependent on the specific application needs and the pads implemented on the device.

The required controls are:

- Slew rate control
  - This is needed to offer improved EMC performance.
  - Support for a fast and a slow slew rate.
- Drive strength control
  - This is needed to offer improved EMC performance.
  - Up to 4 different drive strength levels can be supported.
- Pull capability
  - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
  - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Input and Out Buffer control
- Hysteresis
- Inversion
- Open drain
  - Needed to support different pads muxed (e.g. mux I^2C, with non open drain pads)
- Pad Output Assignment
  - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

12.3.3 General Purpose Input and Output Pads (GPIO)

The SIUL2 allows each pad to be configured as either a General Purpose Input or Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage up to 512 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in the following figure, all port access are identical with each read or write being performed only at a different location to access a different port width.

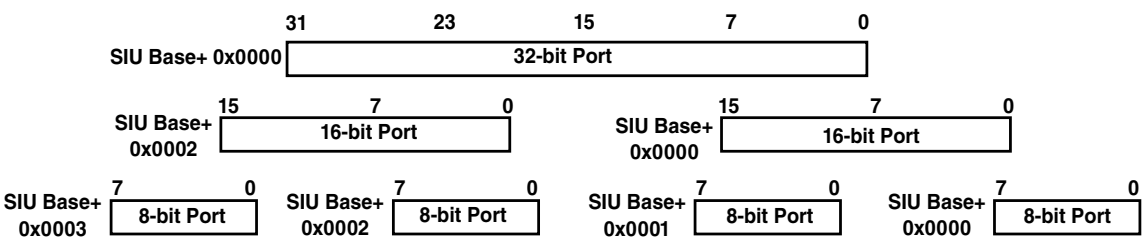


Figure 12-4. Data Port example arrangement showing configuration for different port width accesses

This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until re-configured to GPIO. All general purpose pads are implemented as bidirectional.

Note

In case the bi-directional operation impacts some performances (e.g. ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only"

12.3.4 Alternative pad functions

Module outputs are controlled by the SIUL2_MSCR[SSS] bit. If an output is required for a specific module (such as DSPI), then SSS should be selected to enable the module to output to the pad. For all modules (except the I2C module - the I2C module requires the IBE bit and ODE bit in the MSCR to also be set, since the protocol connections are bi-directional.) this is all that is required: when SSS is set the module selected correctly configures the pin. See the Pin Muxing spreadsheet attachment for details of the alternate functions and SSS settings for each pad.

Module inputs are controlled by the SIUL2_MSCR[IBE] bit, and associated IMCR register. If the function is for the module to be an input, then IBE and the SIUL2_IMCR[SSS] bit needs to be programmed by the user. See the Pin Muxing spreadsheet attachment for details of the alternate functions and the associated IMCR register and SSS bit settings for each pad.

12.3.5 External Interrupts/DMA Requests (REQ Pins)

The SIUL2 supports 1 to 32 external interrupts, which can be allocated to any pad necessary at the MCU level. This allocation is fixed per MCU.

For the assignments of the external interrupts, see the Signal Description details.

The SIUL2 supports 1 to 4 interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support up to 8 external interrupt sources from the device pads.

For devices with a number of interrupt sources that is smaller than the maximum number, the MCU can implement either all four interrupt vectors with fewer sources or reduce the number of vectors as the number of sources are reduced. The following figure shows two sample scenarios.

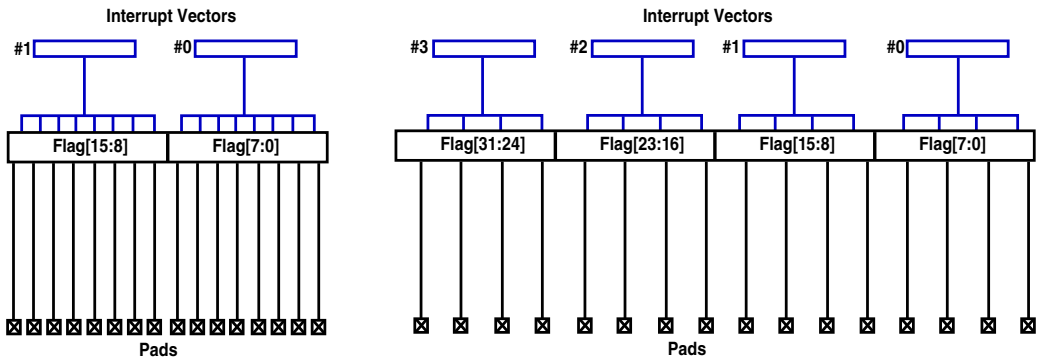


Figure 12-5. Example of different interrupt to vector mapping at MCU level

All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is 1 to 32. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on a interrupt line.

12.3.5.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0:199] register for the external interrupt pin(s) desired as follows:
  - Clear the OBE and ODE bits to disable output.
  - Set the IBE bit to enable the pin's input buffer.
  - If using the internal pullup/pulldown, configure the appropriate PUE and PUS fields.

Note

MSCR[SSS] bits do not need to be changed for either MSCR[0:199] or IMCR[0:199] because the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR[ODC] bits are not zeros) when external interrupt inputs are desired because false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
- Select the desired glitch filter setup for the pins by writing the following:
  - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
  - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
  - Enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.
- Write to EIFn bits in DISR0 to as desired to clear any flags.
- Enable the interrupt pins by setting the appropriate EIREn bits in DIRER0.

12.3.5.2 External interrupt management

The user can enable or disable each interrupt independently using a single rolled up register, SIUL2_DIRER0.

A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge, or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. The user controls the active IRQ edge through the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

Refer to the following figure for an overview of the external interrupt implementation.

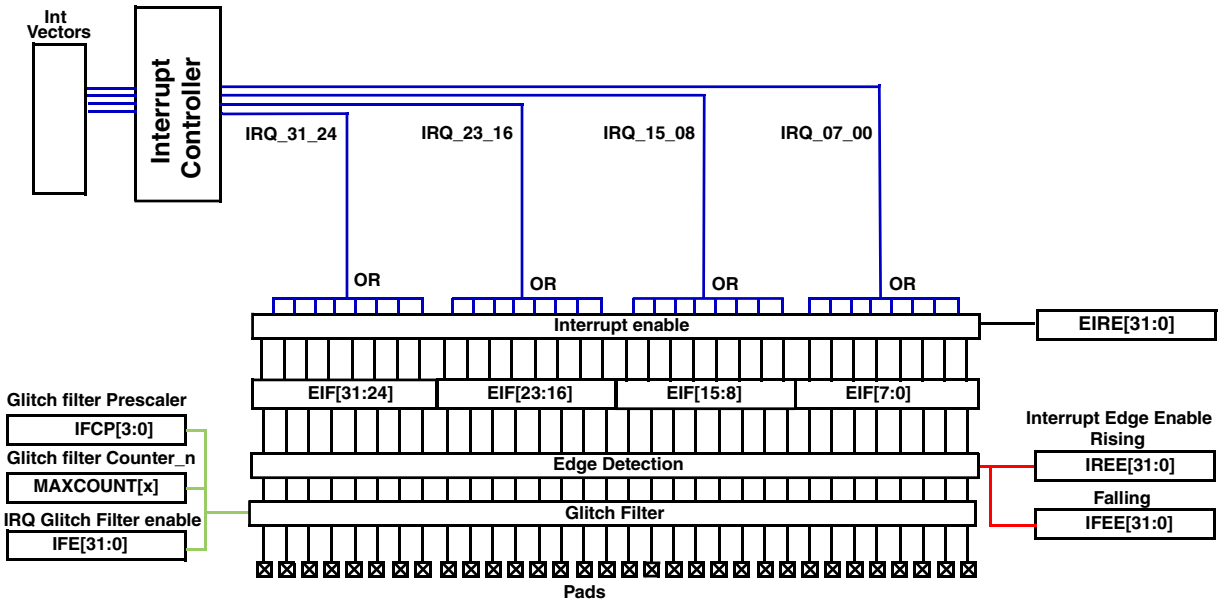


Figure 12-6. External interrupt pad diagram

12.3.5.3 External Interrupt request

The REQ input pins on the device are sources for interrupt or DMA requests. The system has four possible interrupt vectors available for the REQ pins in the SIUL2. The mapping of 32 interrupt request sources to vectors and channels appears in the following table.

Table 12-5. Interrupt source mapping to SIUL2 Interrupt request output for 32 interrupt sources

Vector/Channel #	Interrupt Vector Source
0	REQ[07] | REQ[06] | REQ[05] | REQ[04] | REQ[03] | REQ[02] | REQ[01] | REQ[00]
1	REQ[15] | REQ[14] | REQ[13] | REQ[12] | REQ[11] | REQ[10] | REQ[09] | REQ[08]
2	REQ[23] | REQ[22] | REQ[21] | REQ[20] | REQ[19] | REQ[18] | REQ[17] | REQ[16]
3	REQ[31] | REQ[30] | REQ[29] | REQ[28] | REQ[27] | REQ[26] | REQ[25] | REQ[24]

Chapter 13
Core Complex Overview

13.1 Core Complex Overview

The device has four separate cores that perform various computational and control functions. The main computational shell consists of Main Core_0 that uses e200z4201n3 core, and Main Core_1 and Main Core_2, both of which use an e200z7260n3 core. These three cores are used for general computational functions. A fourth core, referred to as Checker Core_0s, uses an e200z419 core, which is a true subset of the e200z4201n3 core. When enabled, Checker Core_0s operates in Lock Step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Checker Core_0s checks to ensure that Main Core_0 is executing correctly. There is no Lock Step function associated with Main Core_1 and Main Core_2. The e200z419 core, used as Checker Core_0s, differs in hardware from the e200z4201n3 in that it has no Instruction Cache, Data Cache, I-memory, or D-memory.

This chapter provides overviews of the e200z4201n3 core, the e200z419 core, and the e200z7260n3 core.

13.2 Overview of the e200z4201n3, e200z419, and e200z7260n3 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power Architecture.

Each core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z4201n3, e200z419, and e200z7260n3 are dual-issue, 32-bit Power VLE (variable-length-encoding) compliant designs with 32-bit/64-bit general-purpose registers (GPRs) for improved code density. The VLE ISA is further documented in PowerISA 2.06, a separate document. The base PowerISA 2.06 fixed-length 32-bit instruction set is not directly supported.

An Embedded Floating-point (EFPU2) Arithmetic Processing Unit is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Signal Processing Extension (SPE2) Arithmetic Processing Unit in e200z7260n3 core is provided to support real-time SIMD fixed point and single-precision, embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs have been extended to 64-bits in order to support vector instructions defined by the SPE2 APU. These instructions operate on vectors of 8-bit, 16-bit or 32-bit data types, and deliver vector and scalar results. A limited set of 64-bit scalar operations are also supported.

The e200z4201n3 core differs in hardware from the e200z7260n3 core in several ways. The e200z4201n3 core and e200z7260n3 core have different amounts of Instruction and Data Cache. e200z4201n3 contains an 8 KB Instruction Cache, and a 4 KB Data Cache, and e200z7260n3 contains a 16 KB Instruction Cache, and a 16 KB Data Cache.

Both e200z4201n3 and e200z7260n3 cores contain Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit is also included which supports protections of various instruction and data memory areas.

13.3 Features

The following table lists some of the key features of the cores supported in this device.

Table 13-1. e200z4201n3, e200z419, and e200z7260n3 Features

Features	e200z4201n3, e200z419	e200z7260n3
Dual issue 32-bit PowerISA 2.06-VLE compliant CPU	Yes	Yes
In-order execution and retirement	Yes	Yes
Precise exception handling	Yes	Yes
Branch processing unit	— Dedicated branch address calculation adder	— Dedicated branch address calculation adder

Table continues on the next page...

Table 13-1. e200z4201n3, e200z419, and e200z7260n3 Features (continued)

Features	e200z4201n3, e200z419	e200z7260n3
	— Branch target prefetching using BTB	— Branch target prefetching using BTB — Return Address Stack
Load/store unit	— 2 cycle load latency — Fully pipelined — Misaligned access support	— 3 cycle load latency — Fully pipelined — Misaligned access support
General Purpose Register file	32-bit	64-bit
Dual AHB 2.v6 64-bit System buses	Yes	Yes
Local Data Memory (DMEM) with shared AHB 2.v6 64-bit slave interface	— e200z4201n3 core—64 KB DMEM — e200z419 core does not contain either Instruction or Data MEM	64 KB DMEM
Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions	Yes	Yes
2-Way Set Associative Harvard Instruction and Data Cache	— e200z4201n3 core—8 KB ICache and 4 KB DCache — e200z419 core does not contain either Instruction or Data Cache	16 KB ICache and 16 KB DCache
Embedded Floating-point APU (EFPU2) supporting scalar and vector single-precision floating-point operations	Yes	Yes
Signal Processing Extension (SPE2) APU supporting SIMD fixed-point operations, using the 64-bit General Purpose Register file.	No	Yes
Performance Monitor APU supporting execution profiling	Yes	Yes
Nexus Class 3+ Real-time Development Unit	— e200z4201n3 core—Yes — e200z419—No	Yes
Power management — Low power design - extensive clock gating — Power saving modes: DOZE, NAP, SLEEP, WAIT — Dynamic power management of execution units, Caches and Local memory	Yes	Yes
Testability — Synthesizeable, MuxD scan design — ABIST/MBIST for arrays — Built-in LBIST Unit	Yes	Yes
Book E support	No	No

13.4 Microarchitecture summary

The e200z4201n3 and e200z419 processor cores utilize a five stage instruction pipeline with two stages for execution and e200z7260n3 processor utilizes a ten stage instruction pipeline, with four stages for execution. The Instruction Fetch, Instruction Decode/ Register File Read/ EA Calc, Execute 0/ Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single clock instruction execution for most instructions. The following figure shows a block diagram of the e200z architecture.

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. Integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clocks (in z4 core) and four clocks (in z7 core) for correctly predicted branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The SPE2 APU in e200z7260n3 supports vector instructions operating on 8, 16-, and 32-bit fixed-point data types, and scalar instructions operating on 64-bit fixed-point data types.

The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats, supports scalar and vector single-precision floating-point operations, and operates in a pipelined fashion. The general purpose register file is used for source and destination operands, and a unified storage model is used for scalar single-precision floating-point data types of 32-bits and the normal integer type. Low latency fixed-point and floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided, and most operations can be pipelined.

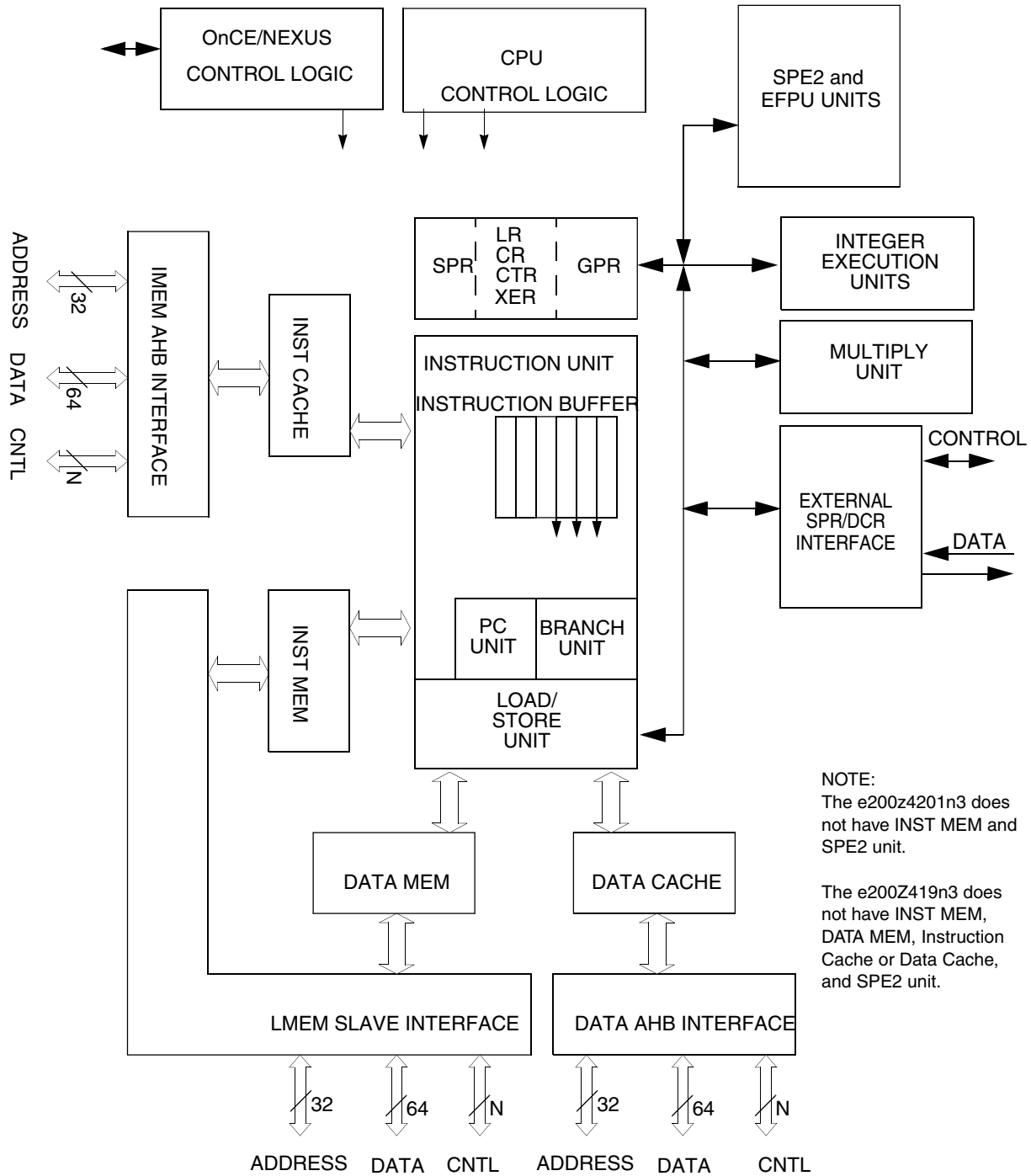


Figure 13-1. e200z block diagram

13.5 Instruction Unit Features

The features of the Instruction unit are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to 8 32-bit instructions in e200z4201n3 and e200z419, and 10 32-bit instructions in e200z7260n3

- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder, and branch lookahead logic (BTB) supporting single cycle execution of successfully predicted branches

13.6 Integer Unit Features

The integer units support single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for count leading zero's function
- 32-bit single cycle barrel shifter for shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-15 clocks (z4 core) and 4-14 clocks (z7 core) with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

13.7 Load/Store Unit Features

The load/store unit supports load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions
- Pipelined operation supports throughput of one load or store operation per cycle

13.8 EFPU2 Floating-Point Unit Features

The EFPU2 embedded floating-point unit supports pipelined operation of most floating-point operations, allowing a throughput of one FP operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion <->single-precision format
- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined scalar floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput

- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results
- Floating-point Divide and Square root logic operating in 13 (FP div) and 11 (FP sqrt) clocks

13.9 SPE2 Signal Processing Unit Features

The SPE2 APU in e200z7260n3 is designed to accelerate signal processing applications normally suited to DSP operation.

- SPE2 instructions operate on the 64-bit GPRs
- Supports SIMD integer operations - arithmetic, logical, shift
- Supports SIMD rounding, saturation, pack, unpack, merge, extract, select for efficient data manipulation
- Supports SIMD multiply/ multiply accumulate/ dot product operations
- 8, 16-, 32-, and 64-bit signed and unsigned integer data types
- 16-, 32-, and 64-bit signed fractional support (1.15, 1.31, 1.63 representations)
- Guarded signed fractional support (33.31 and 17.47 representations)
- Single-cycle throughput for all instructions (except divide)
- Supports fractional divide operations (32/32->32 signed/unsigned int/frac, 64/64->64 signed/unsigned int)
- Specialized operations such as field manipulation, count-leading,
- Supports specialized load/store operations with integrated data manipulation
- Multiple addressing modes supported including circular and bit-reversed addressing
- Sustained throughput of up to 4 16x16-bit multiply/accumulate operations per cycle.

13.10 Memory Protection Unit (MPU) Features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support up to 4 GB
- 8-bit Process Identifier
- Bypass capability for individual access types
- Maskable Address and Process Identifier
- Programmable Memory Attributes
- Entry Invalidation Protection
- Per-Entry Write-Once Option
- Alternate Debug Breakpoint/Watchpoint function

13.11 Cache Features

The features of the Caches are as follows:

- 2-Way Set Associative Instruction and Data Caches
  - e200z4201n3—8 KB ICache, 4 KB DCache
  - e200z419—No ICache, No DCache
  - e200z7260n3—16 KB ICache, 16 KB DCache
- Writethrough Support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Supports Way locking
- Supports Write allocation policies
- Supports multi-bit EDC with Correction/Auto-invalidation capability
- Hardware Debug Cache Invalidate Support for the Data Cache
- Software and Hardware Debug access to cache tag, status and data storage via dedicated control registers

13.12 Local Memory Features

The features of the Local Data Memory are as follows:

- Local Data Memories (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z4201n3 and e200z7260n3—64 KB DMEM
  - e200z419—No DMEM
- Two waitstate (z7) and one waitstate (z4) slave DMEM Read access
- Two waitstate (z7) and one waitstate (z4) slave DMEM Write access
- 64-bit local bus to CPU supporting zero wait-state CPU access
- Supports multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control

13.13 Performance Monitor Unit Features

The features of the e200z4201n3 and e200z7260n3 performance monitor include:

- Four configurable 32-bit performance monitor counters each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use

- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events
- Multiple selectable event types including # of clocks, CPU stall cycles, # of instructions completed, # of interrupts taken, # of memory accesses (by type), # of instruction or data cache misses, # of cycles with 0, 1, or 2 instructions issued, # of instructions in a given class completed (ldst, branch, integer, FP, etc.), cache linefills, and many other event types.

13.14 e200z4201n3 and e200z7260n3 system bus features

The features of the e200z4201n3 and e200z7260n3 System Bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

13.15 e200z419 system features

Checker Core_0 is implemented using an e200z419 core. When enabled, Checker Core_0s is used in Lock Step with Main_Core_0. The e200z419 core can only be enabled or disabled by the user. For this device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z419 core is identical to the e200z4201n3 with five exceptions:

- I MEM memory array is not present
- I CACHE memory array is not present
- D MEM memory array is not present
- D CACHE memory array is not present
- The e200z419 has a pair of system bus interfaces, which are driven/sampled by the delayed lockstep logic

13.16 Special Purpose Register Summary

PowerISA 2.06 and implementation-specific SPRs for the cores are listed in the following table. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the mfspr and mtspr instructions. In the instruction syntax, compilers should recognize the mnemonic name given in the following table.

Table 13-2. Special Purpose Registers

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/ Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/ Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear[1]	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No

Table continues on the next page...

Table 13-2. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1[2]	318	R/W	Yes	No
DVC2	Data Value Compare 2[2]	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes

Table continues on the next page...

Table 13-2. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear[3]	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U[4]	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U[4]	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes

Table continues on the next page...

Table 13-2. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR[5]	SPT Event	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using mfspr RT,DBSR. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to ‘1’ bits in GPR(RS) can be cleared using mtspr DBSR, RS.
2. Undefined on POR assertion.
3. The Machine Check Syndrome Register can be read using mfspr RT, MCSR. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to ‘1’ bits in GPR(RS) can be cleared using mtspr MCSR, RS.
4. The 64-bit DVC registers are accessed by using the DVC1, 2 spr #s for the lower 32 bits, and the DVC1, 2U spr #s for the upper 32 bits.
5. The System Version Register (SVR) is a read-only special purpose register (1023) available to the z7 cores. Each core has an SVR. The SVRs directly mirror registers within the SPT module: Core 1 (z7a) SVR mirrors Core 1 Version Register Events (SPT_CORE1_VER_EVT) and Core 2 (z7b) SVR mirrors Core 2 Version Register Events (SPT_CORE2_VER_EVT)

13.17 Reset settings

The below table shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 13-3. Reset settings for e200z resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] || 2'b00[1]
GPRs	Unaffected[2]
CR	Unaffected[2]
BUCSR	0x0000_0000
CSRR0	Unaffected[2]
CSRR1	Unaffected[2]
CTR	Unaffected[2]
DAC1	0x0000_0000[3]
DAC2	0x0000_0000[3]
DAC3	0x0000_0000[3]
DAC4	0x0000_0000[3]
DBCR0	0x0000_0000[3]
DBCR1	0x0000_0000[3]
DBCR2	0x0000_0000[3]
DBCR4	0x0000_0000[3]
DBCR5	0x0000_0000[3]
DBCR6	0x0000_0000[3]
DBCR7	0x0000_0000[3]
DBCR8	0x0000_0000[3]
DBSR	0x1000_0000[3]
DDAM	0x0000_0000[3]
DDEAR	Unaffected[2]
DEAR	Unaffected[2]
DEVENT	0x0000_0000[3]
DSRR0	Unaffected[2]
DSRR1	Unaffected[2]
DVC1U	Unaffected[2]
DVC2U	Unaffected[2]
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000[3]
IAC2	0x0000_0000[3]

Table continues on the next page...

Table 13-3. Reset settings for e200z resources (continued)

Resource	System reset setting
IAC3	0x0000_0000[3]
IAC4	0x0000_0000[3]
IAC5	0x0000_0000[3]
IAC6	0x0000_0000[3]
IAC7	0x0000_0000[3]
IAC8	0x0000_0000[3]
IVPR	Unaffected[2]
LR	Unaffected[2]
L1CFG0, L1CFG1[4]	–
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS0	Unaffected[2]
MAS1	Unaffected[2]
MAS2	Unaffected[2]
MAS3	Unaffected[2]
MCAR	Unaffected[2]
MCSR	0x0000_0000
MCSRR0	Unaffected[2]
MCSRR1	Unaffected[2]
MMUCFG[4]	–
MPU0CSR0	0x0000_0000
MPU0CFG[4]	–
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 || p_cpuid[0:7][1]
PVR[4]	–
SPEFSCR	0x0000_0000
SPRG0	Unaffected[2]
SPRG1	Unaffected[2]
SPRG2	Unaffected[2]
SPRG3	Unaffected[2]
SRR0	Unaffected[2]
SRR1	Unaffected[2]
SVR[4]	–
XER	0x0000_0000

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
2. Undefined on POR assertion.
3. Reset by processor reset external PORESET (EXT_POR) if DBCR0[EDM]=0, as well as unconditionally by POR.
4. Read-only registers.

Chapter 14
Core description

14.1 Overview of the e200z4201n3 core

The e200z4201n3 is a dual-issue 32-bit *Power ISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z4201n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *Power ISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The e200z4201n3 core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z4201n3 core contains an 8 KB Instruction Cache, and a 4 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

14.2 Register model

This section describes the registers implemented in the e200z4201n3 core. The *Power ISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z4201n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 14-1 and Figure 14-2 show the complete e200z4201n3 register set, all of which are accessible in supervisor mode. Figure 14-3 and Figure 14-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

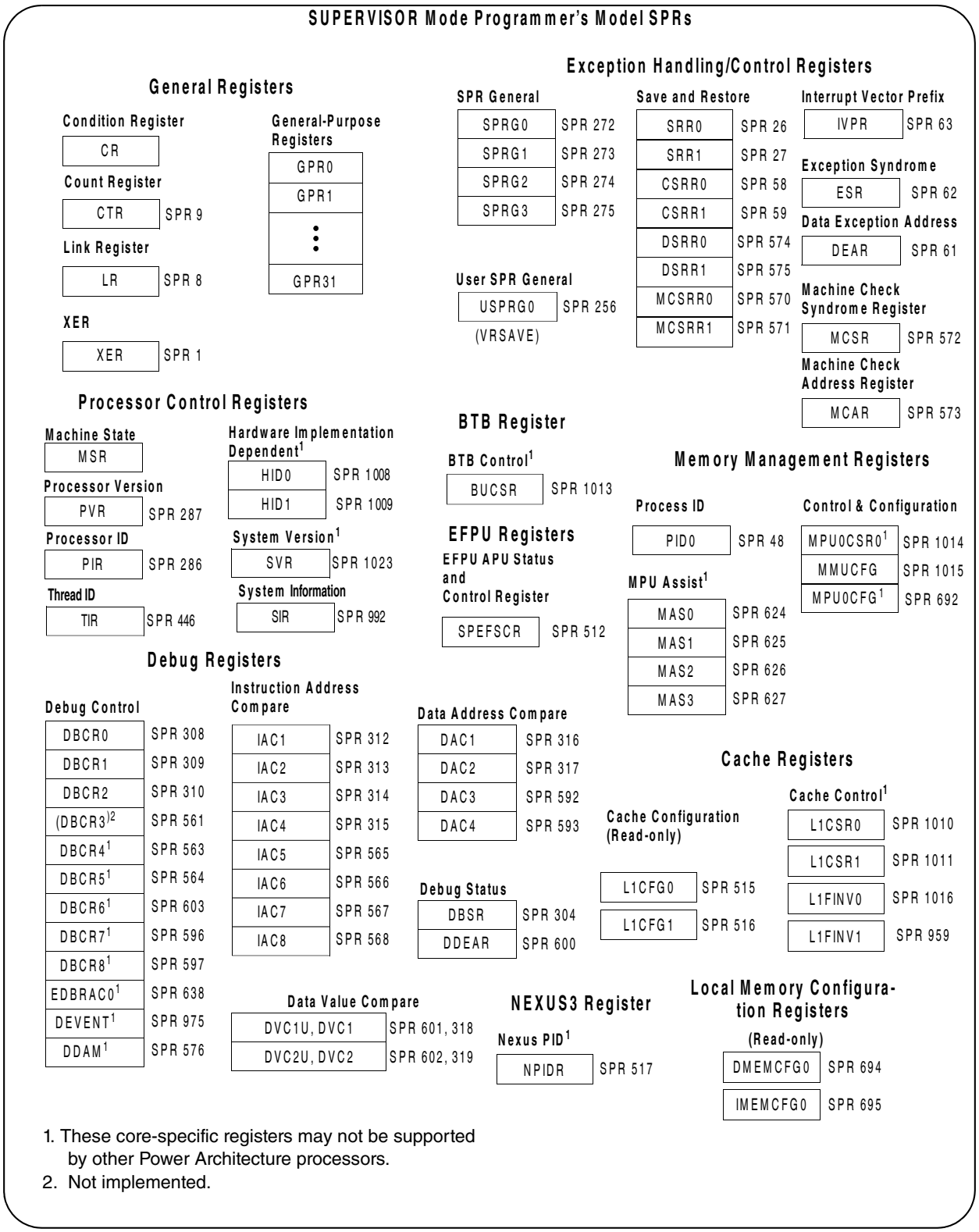


Figure 14-1. e200z4201n3 Supervisor Mode Programmer's Model SPRs

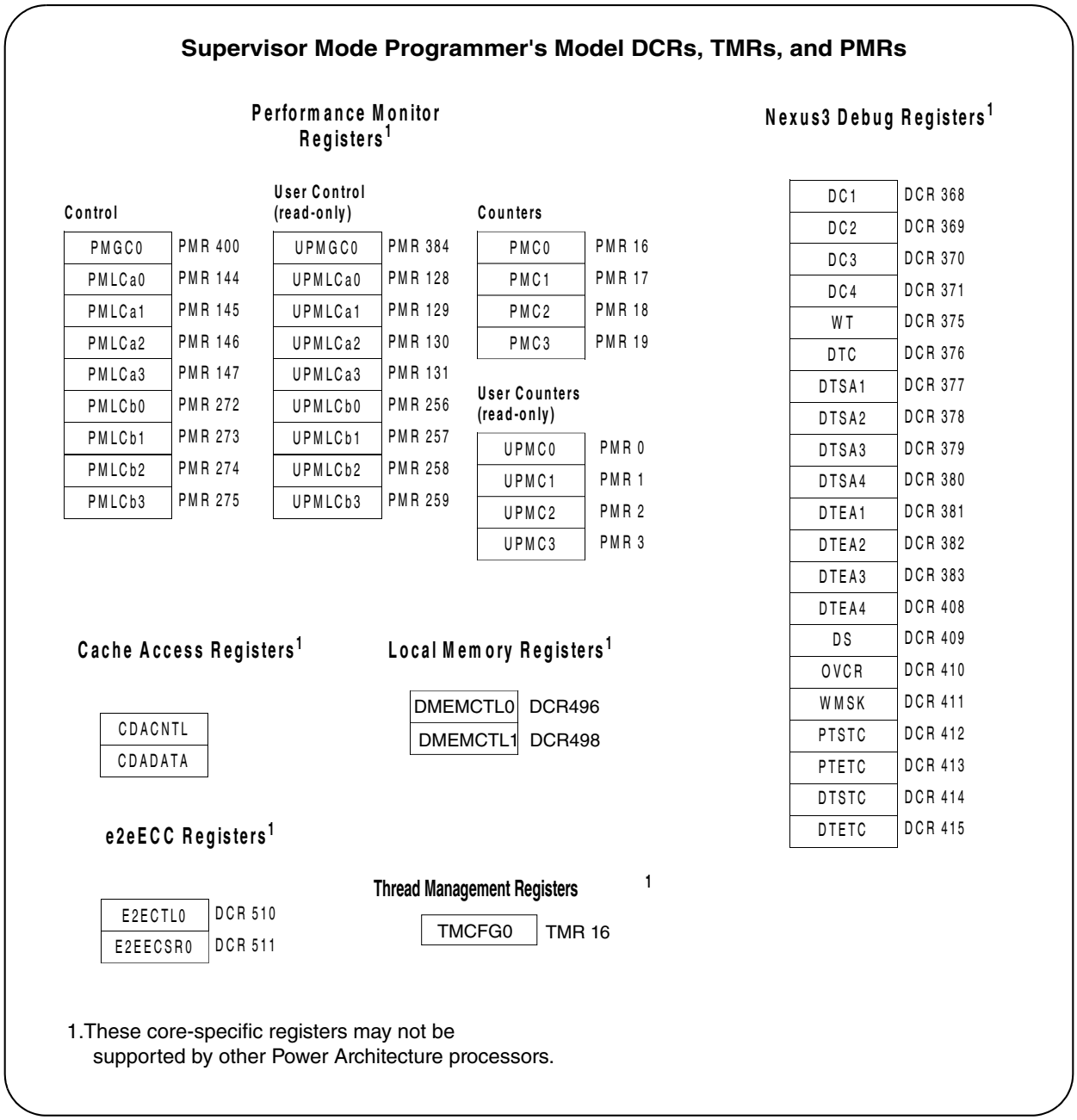


Figure 14-2. e200z4201n3 Supervisor Mode Programmer's Model DCRs and PMRs

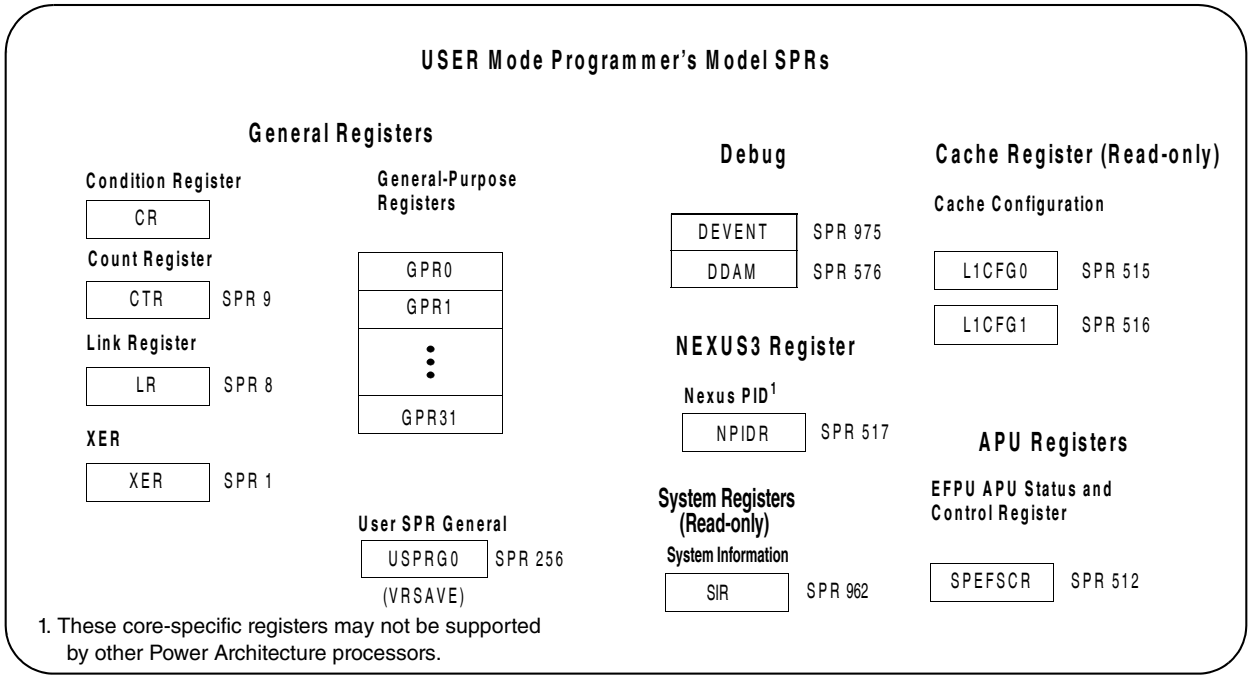


Figure 14-3. e200z4201n3 User Mode Programmer's Model SPRs

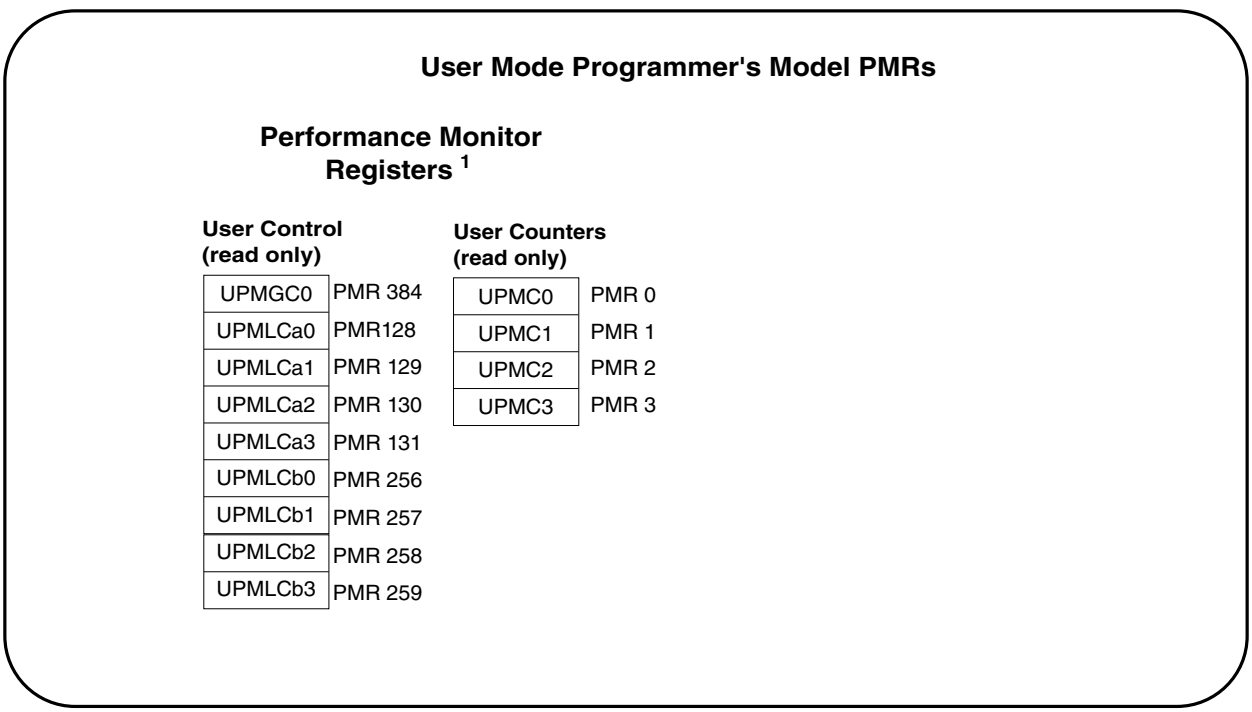
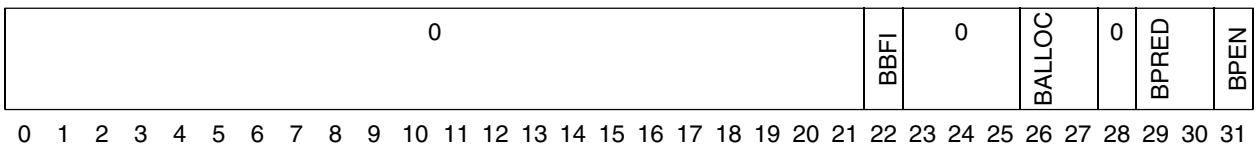


Figure 14-4. e200z4201n3 User Mode Programmer's Model PMRs

14.2.1 Branch Unit Control and Status Register (BUCSR)

The BUCSR register is used for general control and status of the branch target buffer (BTB). The BTB is detailed in Instruction prefetch buffers and branch target buffer. The BUCSR is shown in the following figure.



SPR — 1013; Read/Write; Reset — 0x0

Figure 14-5. Branch Unit Control and Status Register (BUCSR)

The BUCSR fields are defined in the following table.

Table 14-1. Branch Unit Control and Status Register (BUCSR)

Bits	Name	Description
0:21 [32:53]	-	Reserved
22 [54]	BBFI	Branch target buffer flash invalidate. When written to a '1', BBFI flash clears the valid bit of all entries in the branch target buffer; clearing occurs regardless of the value of the enable bit (BPEN). Note: BBFI is always read as 0.
23:25 [55:57]	-	Reserved
26:27 [58:59]	BALLOC	Branch Target Buffer Allocation Control. This field controls BTB allocation for branch acceleration when BPEN = 1. Note that BTB hits are not affected by the settings of this field. 00 Branch Target Buffer allocation for all branches is enabled. 01 Branch Target Buffer allocation is disabled for backward branches. 10 Branch Target Buffer allocation is disabled for forward branches. 11 Branch Target Buffer allocation is disabled for both branch directions.
28 [60]	-	Reserved
29:30 [61:62]	BPRED	Branch Prediction Control (Static). This field controls operation of static prediction mechanism on a BTB miss. When a branch is predicted as taken, fetching of the predicted target location will be performed for branch acceleration. BPRED operates independently of BPEN, and with a BPEN setting of 0, will be used to perform static prediction of all unresolved branches. 00 Branch predicted taken on BTB miss for all branches. 01 Branch predicted taken on BTB miss only for forward branches. 10 Branch predicted taken on BTB miss only for backward branches. 11 Branch predicted not taken on BTB miss for both branch directions. **NOTE:** BPRED functionality is not supported in e200z4xxx processor cores due to its short pipe stages.

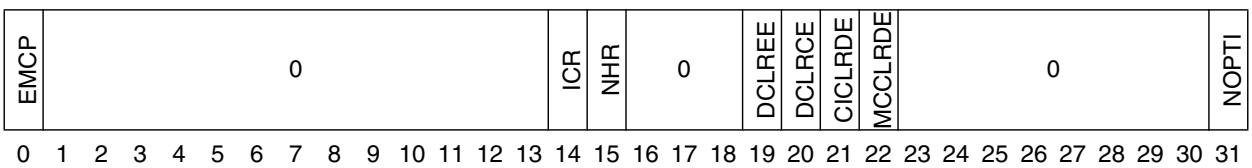
Table continues on the next page...

Table 14-1. Branch Unit Control and Status Register (BUCSR) (continued)

Bits	Name	Description
31 [63]	BPEN	Branch target buffer prediction enable. When the BPEN bit is cleared, no hits will be generated from the BTB, and no new entries will be allocated. Entries are not automatically invalidated when BPEN is cleared; the BBFI bit controls entry invalidation. BPEN operates independently of BPRED, and will be used even with a BPRED setting of 00. 0 Branch target buffer prediction disabled 1 Branch target buffer prediction enabled (enables BTB to predict branches)

14.2.2 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a z4201n3 implementation dependent register used for various configuration and control functions. It is shown in the following figure.



SPR - 1008; Read/Write; Reset - 0x0

Figure 14-6. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 14-2. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit 1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18	-	Reserved

Table continues on the next page...

Table 14-2. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
[48:50]		
19 [51]	DCLREE	Debug Interrupt Clears MSREE 0 MSREEunaffected by Debug Interrupt 1 MSREE cleared by Debug Interrupt This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSRCE 0 MSRCE unaffected by Debug Interrupt 1 MSRCE cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSRDE 0 MSRDE unaffected by Critical class interrupt 1 MSRDE cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit. Note that if Critical Interrupt Debug events are enabled (DBCR0CIRPT set), and MSRDE is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, DSRR0DE will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSRDE enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSRDE 0 MSRDE unaffected by Machine Check interrupt 1 MSRDE cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [58:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

14.2.3 Synchronization requirements for SPRs and DCRs

With the exception of the following registers, there are no synchronization requirements for accessing SPRs and DCRs beyond those stated in PowerISA 2.06. A complete description of Synchronization requirements are contained in Chapter 12 of BookIII-E of PowerISA 2.06. Software requirements for synchronization before/after accessing these

registers are shown in the following table. The notation CSI in the table refers to a Context Synchronizing instruction which include e_sc, se_sc, se_isync, se_rfi, se_rfci, se_rfmci, and se_rfdi.

Table 14-3. Additional synchronization requirements for SPRs and DCRs

Context altering event or instruction		Required before	Required after	Notes
mtmsr[PMM]		none	CSI	
mfspr, mfdcr				
DBSR	Debug Status register	msync	none	
E2EECSR0	End-to-End ECC Error Control and Status Register 0	msync	none	
HID0	Hardware Implementation-Dependent reg 0	none	none	
HID1	Hardware Implementation-Dependent reg 1	msync	none	
L1CSR0, L1CSR1	L1 cache control and status registers 0,1	msync	none	
L1FINV0, L1FINV1	L1 cache flush and invalidate control registers 0,1	msync	none	
MPU0CSR0	MPU control and status register 0	CSI	none	
mtspr, mtdcr				
BUCSR	Branch Unit Control and Status Register	none	CSI	
DBCR0-10	Debug Control Register 0-10	none	CSI	
DBSR	Debug Status Register	msync	none	
DMEMCTL0,1	Local Data Memory Control Register 0 and 1	msync, isync	CSI	
E2ECTL0	End-to-End ECC Control and Status Register 0	msync, isync	CSI	
E2EECSR0	End-to-End ECC Error Control and Status Register 0	msync	CSI	
HID0	Hardware Implementation-Dependent reg 0	CSI	isync	
HID1	Hardware Implementation-Dependent reg 1	msync, isync	CSI	
L1CSR0	L1 cache control and status register 0	msync, isync	CSI	
L1CSR1	L1 cache control and status register 1	msync, isync	CSI	
L1FINV0, L1FINV1	L1 cache flush and invalidate control registers 0,1	msync	CSI	
MASx	MPU MAS registers	none	CSI	
MPU0CSR0	MPU control and status register 0	CSI	CSI	
PID (PID0)	PID0 register	CSI	CSI	
SPEFSCR	SPEFSCR register	none	CSI1	

Notes:

1. not required for status bit clearing, required for altering exception enable or rounding mode bits

14.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. As many as three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

The following table shows the concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 14-4. Concurrent instruction issue capabilities

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Special
Branch	—	yes	yes	yes	—
Load/Store	yes	—	yes	yes	—
Scalar integer	yes	yes	yes[1]	yes[2]	—
Scalar float	yes	yes	yes[2]	—	—
Special	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.
2. Excludes divide/sqrt class instructions occurring in both issue slots

14.4 Instruction prefetch buffers and branch target buffer

An eight-entry instruction prefetch buffer supplies instructions into the Instruction Register (IR) for decoding. Each slot in the prefetch buffer is 32 bits wide, capable of holding a single 32-bit instruction, or a pair of 16-bit instructions.

Instruction prefetches request a 64-bit doubleword and the prefetch buffer is filled with a pair of instructions at a time, except for the case of a change of flow fetch where the target is to the second (odd) word. In that case only a 32-bit prefetch is performed to load the instruction prefetch buffer. This 32-bit fetch may be immediately followed by a 64-bit prefetch to fill Slots 0 and 1 in the event that the branch is resolved to be taken.

In normal sequential execution, instructions are loaded into the IR from prefetch buffer Slot 0 and 1, and as a pair of slots are emptied, they are refilled. Whenever a pair of slots is empty, a prefetch is initiated which fills the earliest empty slot pairs beginning with Slot 0.

If the instruction prefetch buffer empties, instruction issue stalls, and the buffer is refilled. The first returned instruction is forwarded directly to the IR. Open cycles on the memory bus are utilized to keep the buffer full when possible.

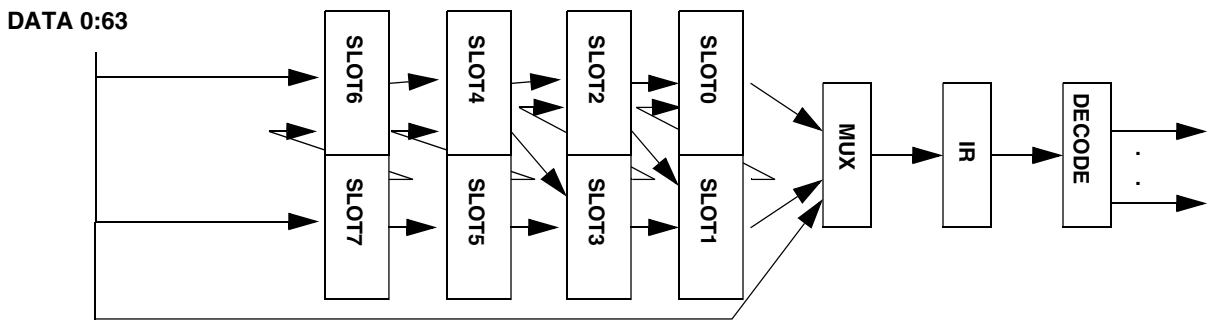


Figure 14-7. Instruction prefetch buffers

To resolve branch instructions and improve the accuracy of branch predictions, the z4201n3 implements a dynamic branch prediction mechanism using an 8-entry branch target buffer (BTB). An entry is allocated in the BTB whenever a branch resolves as taken and the BTB is enabled and misses. Certain branches do not allocate BTB entries: se_bctr, se_bctrl, and se_blrl. Entries in the BTB are allocated on taken branches using a FIFO replacement algorithm.

Each BTB entry holds a branch instruction address tag value, the branch target address, and a 2-bit branch history counter whose value is incremented or decremented on a BTB hit, depending on whether the branch was taken. The counters can assume four different values: strongly taken, weakly taken, weakly not taken, and strongly not taken. On initial allocation of an entry to the BTB for a taken branch, the counter is initialized to the weakly-taken state.

A branch will be predicted as taken on a hit in the BTB with a counter value of strongly or weakly taken. In this case the target address contained in the BTB is used to redirect the instruction fetch stream to the target of the branch prior to the branch reaching the

instruction decode stage. In the case of a BTB miss, static prediction is used to predict the outcome of the branch. In the case of a mispredicted branch, the instruction fetch stream will return to the proper instruction stream after the branch has been resolved.

When a branch is predicted taken and the branch is later resolved (in the branch execute stage), the value of the appropriate BTB counter is updated. If a branch whose counter indicates weakly taken is resolved as taken, the counter increments so that the prediction becomes strongly taken. If the branch resolves as not taken, the prediction changes to weakly not-taken. The counter saturates in the strongly taken states when the prediction is correct.

The z4201n3 does not implement the static branch prediction that is defined by the PowerISA architecture. The BO prediction information in branch encodings is ignored.

Dynamic branch prediction is enabled by setting $BUCSR_{BPEN}$. Allocation of branch target buffer entries may be controlled using the $BUCSR_{BALLOC}$ field to control whether forward or backward branches (or both) are candidates for entry into the BTB, and thus for branch prediction. Once a branch is in the BTB, $BUCSR_{ALLOC}$ has no further effect on that branch entry. Clearing $BUCSR_{BPEN}$ disables dynamic branch prediction, in which case the z4201n3 reverts to a static prediction mechanism using the $BUCSR_{BPRED}$ field to control whether forward or backward branches (or both) are predicted taken or not taken.

The BTB uses instruction fetch addresses for performing tag comparisons. On allocation of a BTB entry, the effective address of a taken branch is loaded into the entry as the tag value, and the counter value is set to weakly taken. The current PID value is not maintained as part of the tag information.

The z4201n3 does support automatic flushing of the BTB when the current PID value is updated by a mtcr PID0 instruction. Software is otherwise responsible for maintaining coherency in the BTB when a change in target address mapping is changed. This is supported by the $BUCSR_{BBFI}$ control bit.

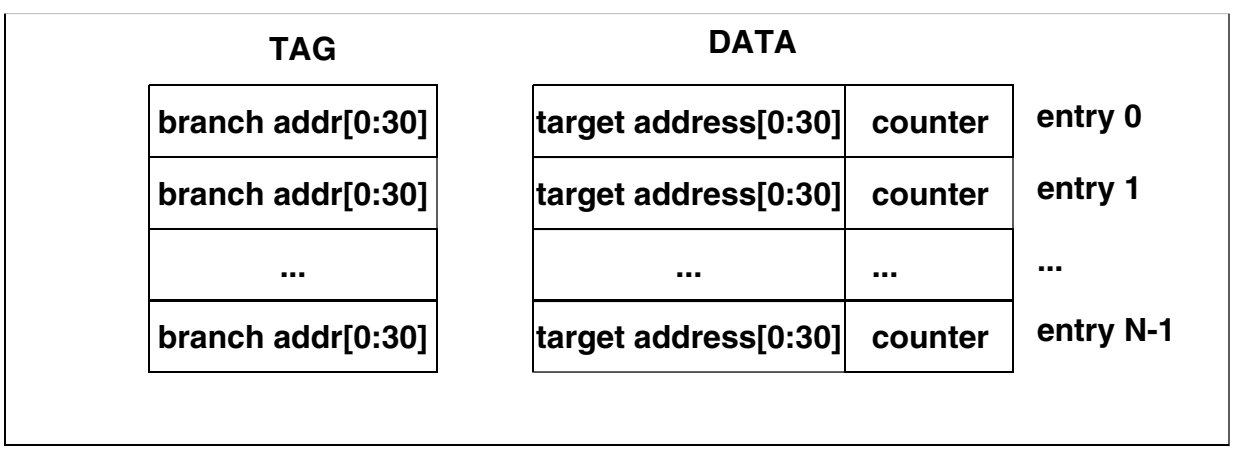


Figure 14-8. Branch target buffer

14.5 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 14-5. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during execution.

Timing for EFPU instructions is detailed in Table 14-8.

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 14-5. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**	1	1	—
Branch	3/2/1	3/2/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 3 cycles.
multiply	2	1	—
divide	4–14	4–14	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	2	1	—
load multiple	2 + *n*/2 (max)	1 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	2	1	—
store multiple	2 + *n*/2 (max)	1 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	3&	3&	—
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	3	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 14-6 and Table 14-7.

Table 14-6. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	3/2/1	—
se_bclri	1	—
se_bctr	3/2	—
se_bctrl	3/2	—
se_bgeni	1	—
se_bl	3/2/1	—
se_blr	3/2	—
se_blrl	3/2	—
se_bmaski	1	—
se_b	3/2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	2	—
se_lhz	2[2]	—
se_li	1	—
se_lwz	2[2]	—
se_mfar	1	—
se_mfctr	1	—

Table continues on the next page...

Table 14-6. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	2	—
se_sth	2[2]	—
se_stw	2[2]	—
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 14-7. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—
addeo[.]	1	—

Table continues on the next page...

Table 14-7. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	3/2/1	—
e_bc	3/2/1	—
e_bcl	3/2/1	—
e_bl	3/2/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—

Table continues on the next page...

Table 14-7. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4–14[1]	—
divwo[.]	4–14[1]	—
divwu[.]	4–14[1]	—
divwuo[.]	4–14[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	2	—
e_lbz	2	—
e_lbzu	2	—
lbzux	2	—
lbzx	2	—
e_lha	2[2]	—
lharx	2	—
e_lhau	2[2]	—
lhaux	2[2]	—
lhax	2[2]	—
lhbrx	2[2]	—
e_lhz	2[2]	—
e_lhzu	2[2]	—
lhzux	2[2]	—
lhzx	2[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2 + (*n*/2)	—
lwarx	2	—
lwbrx	2[2]	—
e_lwz	2[2]	—
e_lwzu	2[2]	—
lwzux	2[2]	—
lwzx	2[2]	—

Table continues on the next page...

Table 14-7. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2	—
mulhwu[.]	2	—
e_mull2i	2	—
e_mulli	2	—
mullw[.]	2	—
mullwo[.]	2	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch
slw[.]	1	—

Table continues on the next page...

Table 14-7. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	2	—
stbcx.	2	—
e_stbu	2	—
stbux	2	—
stbx	2	—
e_sth	2[2]	—
sthbrx	2[2]	—
sthcx.	2	—
e_sthu	2[2]	—
sthux	2[2]	—
sthx	2[2]	—
e_stmw	2 + (*n*/2)	—
e_stw	2[2]	—
stwbrx	2[2]	—
stwcx.	2	—
e_stwu	2[2]	—
stwux	2[2]	—
stwx	2[2]	—
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—

Table continues on the next page...

Table 14-7. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

Instruction timing for EFPU single-precision scalar floating-point instructions is shown in Table 14-8. The table is sorted by opcode.

Table 14-8. EFPU single-precision scalar floating-point instruction timing

Instruction	Latency	Throughput	Comments
efsabs	2	1	—
efsadd	2	1	—
efscfh	2	1	—
efscfsf	2	1	—
efscfsi	2	1	—
efscfuf	2	1	—
efscfui	2	1	—
efscmpeq	2	1	—
efscmpgt	2	1	—
efscmplt	2	1	—
efscth	2	1	—
efsctsf	2	1	—
efsctsi	2	1	—
efsctsiz	2	1	—
efsctuf	2	1	—
efsctui	2	1	—
efsctuiz	2	1	—
efsdiv	13	13	Blocking, no execution overlap with next instruction
efsmadd	2	1[1]	Destination also used as source
efsmsub	2	1[1]	Destination also used as source
efsmax	2	1	—
efsmin	2	1	—
efsmul	2	1	—
efsnabs	2	1	—
efsneg	2	1	—
efsnmadd	2	1[1]	Destination also used as source
efsnmsub	2	1[1]	Destination also used as source
efssqrt	11	11	Blocking, no overlap with next instruction
efssub	2	1	—

Table continues on the next page...

Table 14-8. EFPU single-precision scalar floating-point instruction timing (continued)

Instruction	Latency	Throughput	Comments
efststeq	2	1	—
efststgt	2	1	—
efststlt	2	1	—

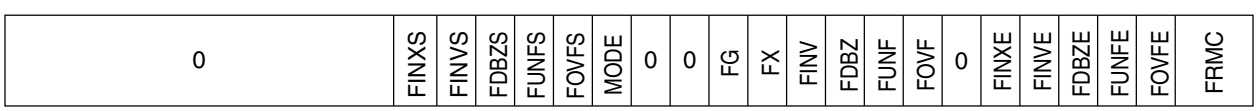
1. Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.

14.6 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

14.7 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.



SPR - 512; Read/Write; Reset - 0x0

Figure 14-9. EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 14-9. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10	FINXS	Embedded Floating-point Inexact Sticky Flag

Table continues on the next page...

Table 14-9. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
(42)		The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z4201n3 Supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by e200 core)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.

Table continues on the next page...

Table 14-9. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

14.8 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

14.8.1 Cache overview

The e200z4201n3 processor supports an 8 KB 2-way set-associative instruction and 4KB 2-way set-associative data cache with a 32-byte line size. The caches improves system performance by providing low-latency data to the e200z4201n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z4201n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

14.8.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in <u>both</u> caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.

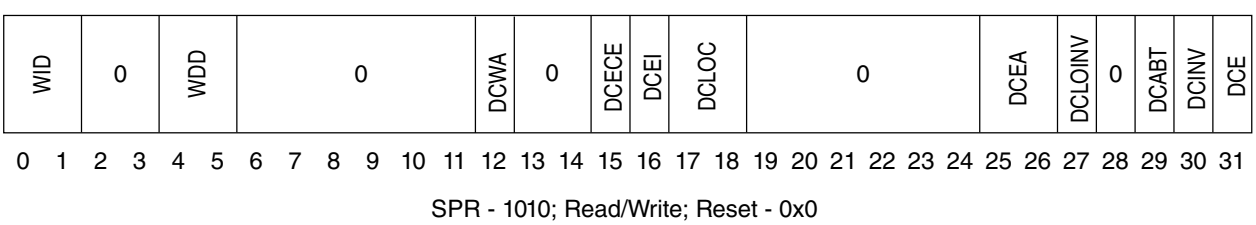


Figure 14-10. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 14-10. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable

Table continues on the next page...

Table 14-10. L1CSR0 field descriptions (continued)

Bits	Name	Description
		Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved[1]
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
8:11	—	Reserved[1]
12	DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved[1]
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18	DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled.

Table continues on the next page...

Table 14-10. L1CSR0 field descriptions (continued)

Bits	Name	Description
		Note: When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table continues on the next page...

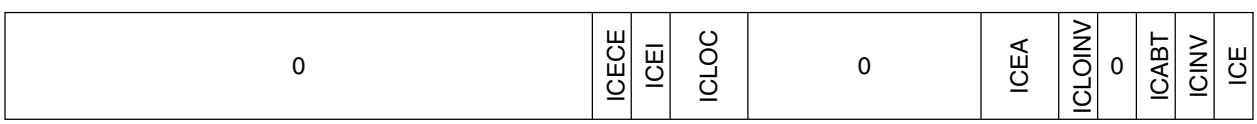
Table 14-10. L1CSR0 field descriptions (continued)

Bits	Name	Description
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses.Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

14.8.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.



SPR - 1011; Read/Write; Reset - 0x0

Figure 14-11. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 14-11. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved[1]
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable

Table continues on the next page...

Table 14-11. L1CSR1 field descriptions (continued)

Bits	Name	Description
		ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation

Table continues on the next page...

Table 14-11. L1CSR1 field descriptions (continued)

Bits	Name	Description
		operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

14.8.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z4201n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

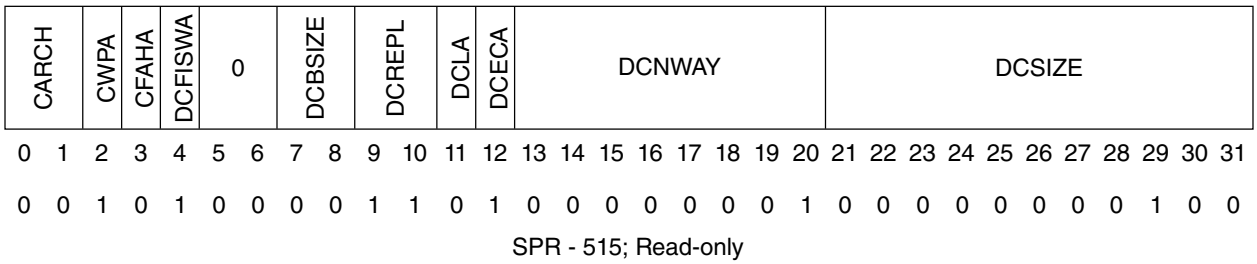


Figure 14-12. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in the following table.

Table 14-12. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

14.8.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z4201n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

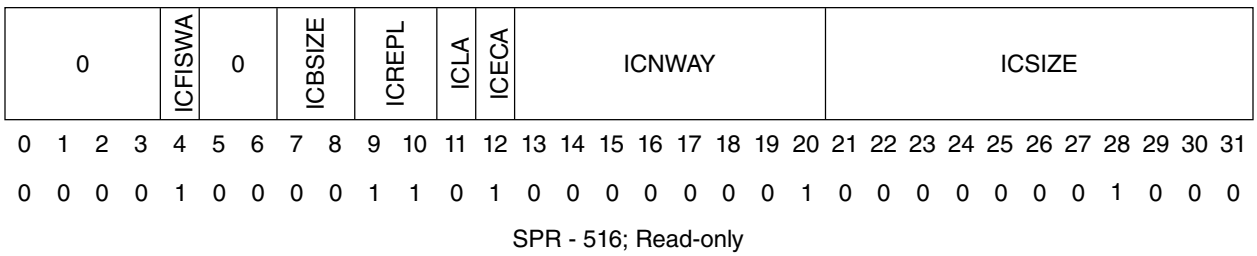


Figure 14-13. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 14-13. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

14.8.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

14.8.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

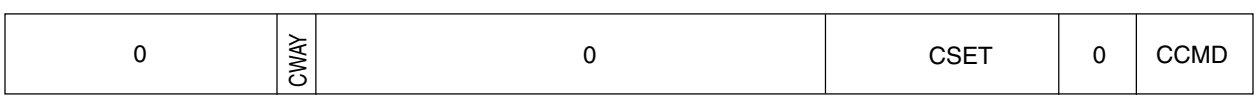
14.8.8 Cache Invalidate by Set and Way

e200z4201n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

14.8.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in the following figure.



SPR - 1016; Read/Write; Reset - 0x0

Figure 14-14. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in the following table.

Table 14-14. L1FINV0 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected

Table continues on the next page...

Table 14-14. L1FINV0 field descriptions (continued)

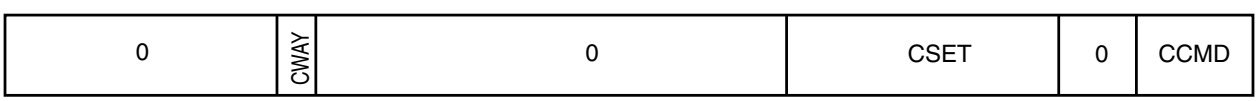
Bits	Name	Description
8: 20	—	Reserved[1] for set extension
21 :26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 = The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

14.8.8.2 L1FINV1

The L1FINV1 register is shown in following figure.



SPR - 959; Read/Write; Reset - 0x0

Figure 14-15. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 14-15. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8: 19	—	Reserved[1] for set extension
20 :26	CSET	Cache Set

Table continues on the next page...

Table 14-15. L1FINV1 field descriptions (continued)

Bits	Name	Description
		Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

14.8.9 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR0[DCECE] and L1CSR1[ICECE] control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache ($L1CSR\{0,1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

14.8.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR\{0,1\}_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

14.8.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If $L1CSR0_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction or an L1FINV{0,1} invalidation operation.

14.9 Exceptions

Interrupts implemented in e200z4201n3 and the exception conditions that cause them are listed in the following table.

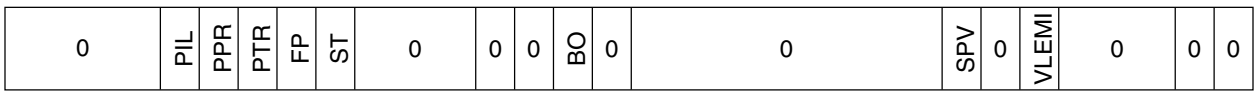
Table 14-16. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

14.9.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 14-16. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 14-17. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z4201n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage

Table continues on the next page...

Table 14-17. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
			Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

14.9.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z4201n3 MSR is shown in the following figure.

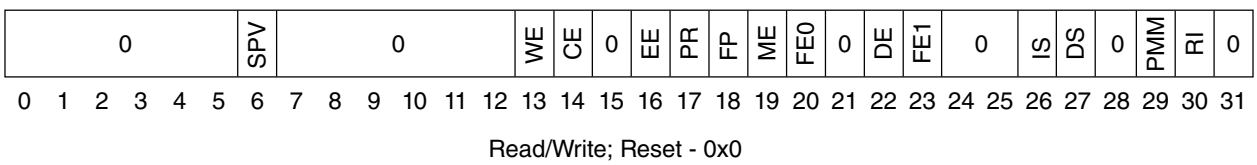


Figure 14-17. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 14-18. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/Embedded FP Double or Vector Unit instructions. **NOTE:** The e200z4201n3 processor does not support these units in hardware. An Illegal Instruction exception is generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** this bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.

Table continues on the next page...

Table 14-18. MSR field descriptions (continued)

Bit(s)	Name	Description
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** The e200z4201n3 processor does not support the PowerISA 2.06 floating point unit in hardware, and an Illegal Instruction exception is generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used) **NOTE:** The e200z4201n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used) **NOTE:** The e200z4201n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** The e200z4201n3 processor does not support Address Spaces, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1.

Table continues on the next page...

Table 14-18. MSR field descriptions (continued)

Bit(s)	Name	Description
		NOTE: The e200z4201n3 processor does not support Address Spaces, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

14.9.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.

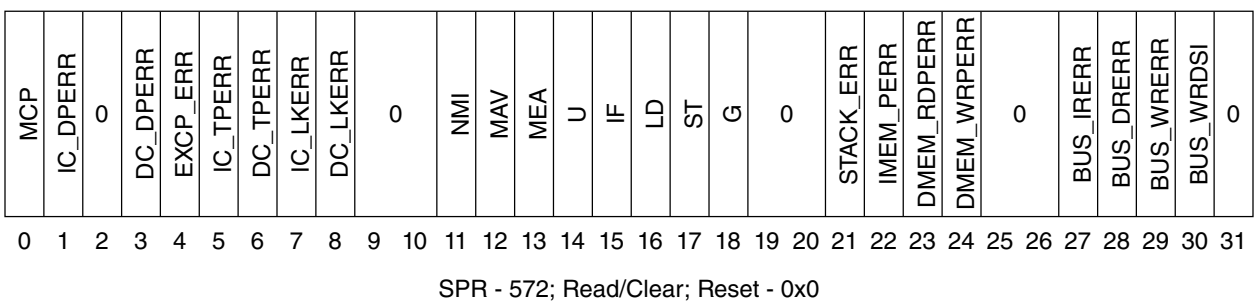


Figure 14-18. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 14-19. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[2]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise

Table continues on the next page...

Table 14-19. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR[3]	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

- "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
- "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

2. Note that since no address translation mechanism is present on e200z4201n3, the MEA value is always set to '0'.
3. Will not be set by e200z4201n3 since no local instruction memory is present.

14.9.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

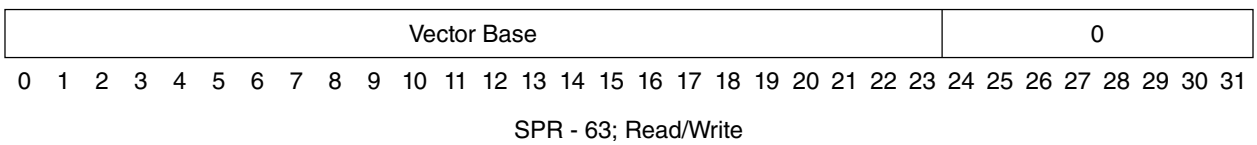


Figure 14-19. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 14-20. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

14.9.5 Interrupt Definitions

14.9.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 14-21. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

14.9.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 14-22. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

14.9.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 14-23. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

14.9.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 14-24. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

14.9.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 14-25. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 14-25. External Input Interrupt—Register Settings (continued)

Register	Setting description					
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x40 (autovectored)					
	IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

14.9.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

The following table lists register settings when an alignment interrupt is taken.

Table 14-26. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

14.9.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z4201n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z4201n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z4201n3 will invoke a Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *Power ISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 14-27. Program Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the excepting instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 14-27. Program Interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

14.9.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 14-28. Performance Monitor Interrupt—Register Settings

Register	Setting description
SRR0/ DSRR0[1]	Set to the effective address of the next instruction to be executed.
SRR1/	Set to the contents of the MSR at the time of the interrupt.

Table continues on the next page...

Table 14-28. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting description					
DSRR1[1]						
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

14.9.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 14-29. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

14.9.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM} = 1$ (internal debug mode) and $MSR_{DE} = 1$).

The following table lists register settings when a Debug interrupt is taken.

Table 14-30. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			
	External Debug Event:		{DEVT1, DEVT2}			
	Performance Monitor Debug Event:		PMI			
	MPU Debug Event:		MPU			
	and optionally, an Imprecise Debug Event flag		{IDE}			
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x90					

1. Assumes that the Debug interrupt is precise

2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

14.9.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 14-31. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

14.9.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 14-32. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 14-32. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description					
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

14.9.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 14-33. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

14.10 Memory Protection Unit (MPU)

14.10.1 MPU Overview

The e200z4201n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

14.10.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

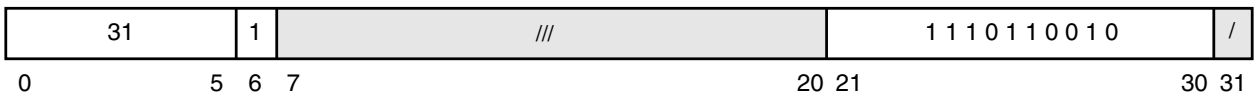
14.10.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

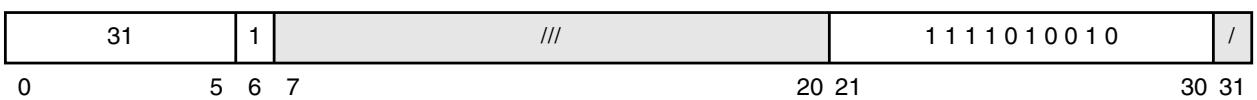
14.10.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

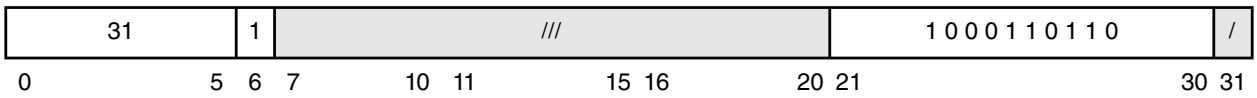
14.10.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the core.

mpusync

MPU Synchronize
mpusync

mpusync



14.10.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z4201n3 MPU design. The MMUCFG register is shown in the following figure.

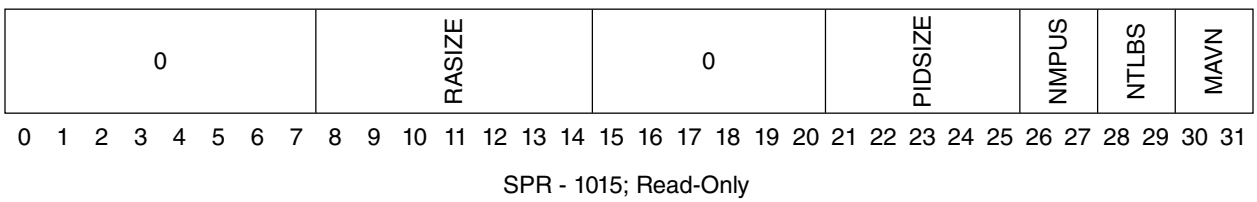


Figure 14-20. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 14-34. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

14.10.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z4201n3 MPU. The MPU0CFG register is shown in following figure.

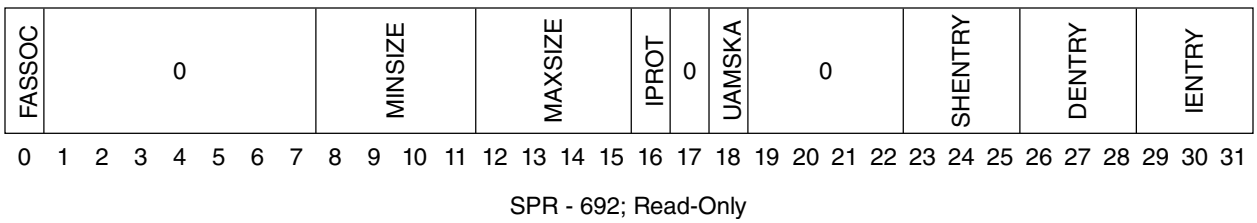


Figure 14-21. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in the following table.

Table 14-35. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

14.10.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.

SPR - 1014; Read/ Write; Reset - 0x0[1]

Figure 14-22. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM} = 0$, as well as unconditionally by an internal power-on reset signal. If $EDBCR0_{EDM} = 1$, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 14-36. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses

Table continues on the next page...

Table 14-36. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses

Table continues on the next page...

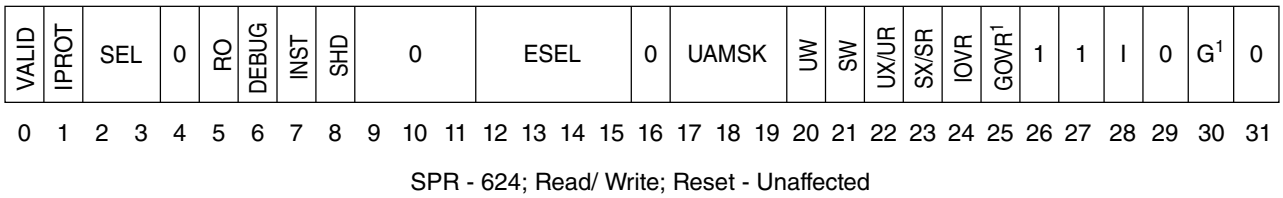
Table 14-36. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

14.10.9 MPU Assist Registers (MAS)

The core uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 14-23. Fields are defined in Table 14-37.



SPR - 624; Read/ Write; Reset - Unaffected

Figure 14-23. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 14-37. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 = This MPU entry is invalid 1 = This MPU entry is valid
1	IPROT	Invalidation Protect 0 = Entry is not protected from invalidation 1 = Entry is protected from invalidation
2:3	SEL	Selects MPU for access: 00 = Reserved, no access 01 = Reserved, no access 10 = Select MPU 11 = Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 = This MPU entry is writable 1 = This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 = This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 = This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations.

Table continues on the next page...

Table 14-37. MAS0—MPU Read/Write and Replacement Control (continued)

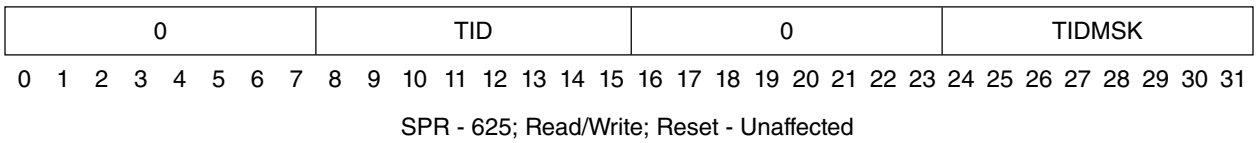
Bit	Name	Comments, or Function when Set
		When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 = This MPU entry is used for matching data accesses only 1 = This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 = The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 = The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12 :15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 = No upper address bits are masked, address matching uses all 32 address bits for accesses 001 = The most significant access address bit is forced to zero before matching 010 = The two most significant access address bits are forced to zero before matching 011 = The three most significant access address bits are forced to 0 before matching 100 = The four most significant access address bits are forced to 0 before matching 101 = The upper five access address bits are forced to zero before matching 11x = Reserved, do not use
20	UW	User Mode Write PermissionDetermines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No User mode Write permission 1 = User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No Supervisor mode Write permission

Table continues on the next page...

Table 14-37. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		1 = Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 = No User mode Execute/ Read permission 1 = User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 = No Supervisor mode Execute/ Read permission 1 = Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute OverrideDetermines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 = No override of I attribute by entry 1 = Entry I bit overrides I attribute
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = No override of G attribute by entry 1 = Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. When the core memory protection unit (CMPU) is disabled, the CMPU configuration is ignored and this bit is ignored also. 0 = This region is considered cacheable 1 = This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings.This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 = All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in Figure 14-24. Fields are defined in Table 14-38.



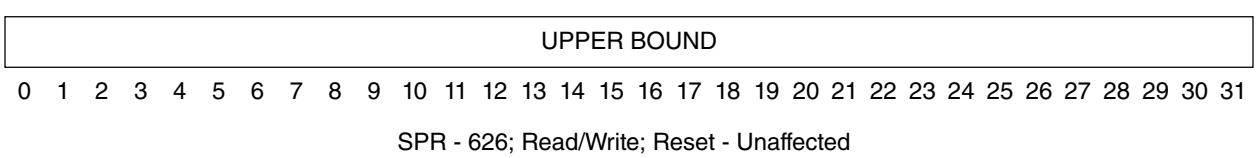
SPR - 625; Read/Write; Reset - Unaffected

Figure 14-24. MPU Assist Register 1 (MAS1)

Table 14-38. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked xx0xxxxx = TID[2] comparison to PID0[2] not masked xx1xxxxx = TID[2] comparison to PID0[2] is masked xxx0xxxx = TID[3] comparison to PID0[3] not masked xxx1xxxx = TID[3] comparison to PID0[3] is masked xxxx0xxx = TID[4] comparison to PID0[4] not masked xxxx1xxx = TID[4] comparison to PID0[4] is masked xxxxx0xx = TID[5] comparison to PID0[5] not masked xxxxx1xx = TID[5] comparison to PID0[5] is masked xxxxxx0x = TID[6] comparison to PID0[6] not masked xxxxxx1x = TID[6] comparison to PID0[6] is masked xxxxxxx0 = TID[7] comparison to PID0[7] not masked xxxxxxx1 = TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 14-25. Fields are defined in Table 14-39.



SPR - 626; Read/Write; Reset - Unaffected

Figure 14-25. MPU Assist Register 2 (MAS2)

Table 14-39. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 14-26. Fields are defined in Table 14-40.

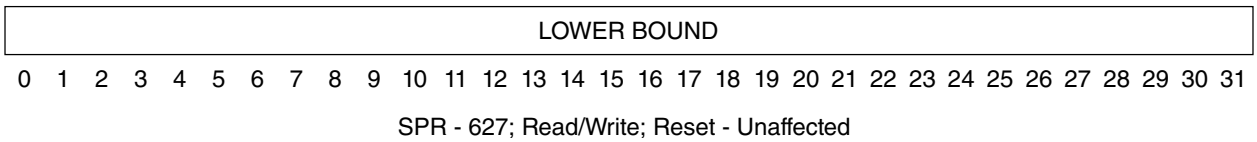


Figure 14-26. MPU Assist Register 3 (MAS3)

Table 14-40. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

14.10.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

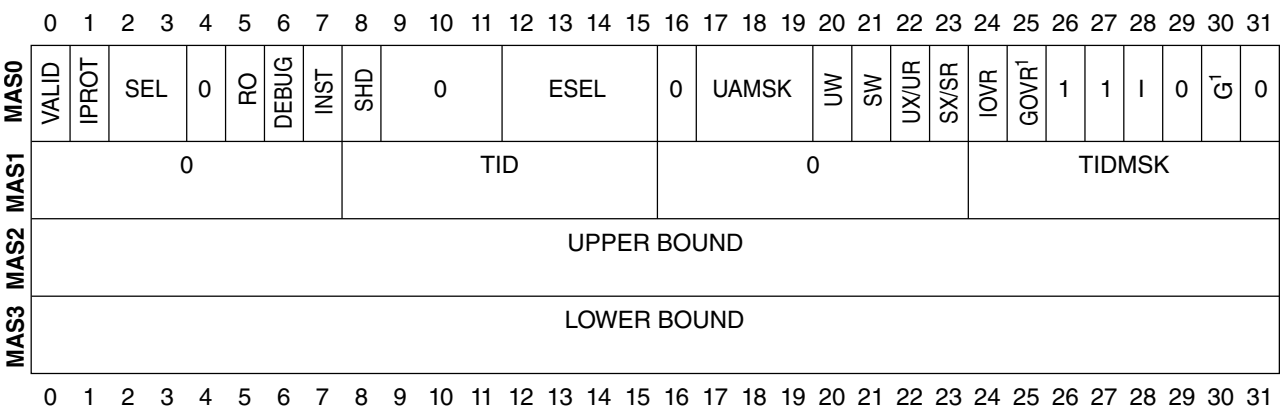


Figure 14-27. MPU Assist Registers Summary

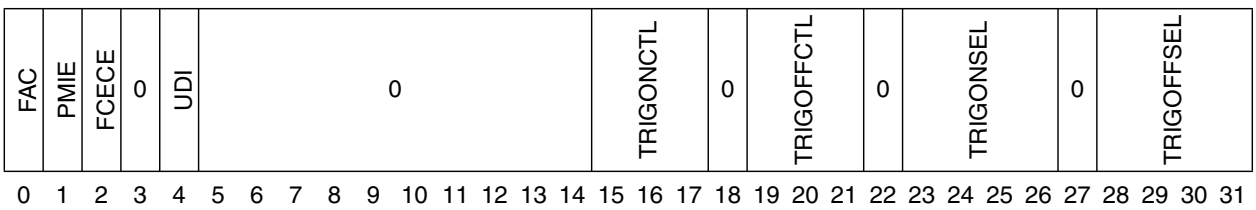
Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

14.11 Performance Monitor

14.11.1 Global Control Register 0 (PMGC0)

The performance monitor global control register PMGC0 shown in the following figure controls all performance monitor counters.



PMR - 400; Read/Write; Reset[1] - 0x0

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 14-28. Performance Monitor Global Control Register (PMGC0)

The following table describes PMGC0 fields.

Table 14-41. PMGC0 field descriptions

Bits	Name	Description
0 (32)	FAC	Freeze All Counters. 0 The PMCs are incremented (if permitted by other PMGC/PMLC control bits). 1 The PMCs are not incremented. When FAC is set to 1 by hardware or software, it has no effect on PMLCaxFC; PMLCaxFC maintains its current value until changed by software. FAC setting by hardware is controlled by PMGC0FCECE.
1 (33)	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs, at which time PMGC0PMIE is cleared Software can clear PMIE to prevent performance monitor interrupts. Performance monitor interrupts are caused by PMCx counter overflows.
2 (34)	FCECE	Freeze Counters on Enabled Condition or Event An enabled condition or event is defined as one of the following: • When the msb = 1 in PMCx and PMLCaxCE = 1. 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0FAC is set to 1. It is up to software to clear PMGC0FAC to 0.
3 (35)	-	Reserved, should be cleared.

Table continues on the next page...

Table 14-41. PMGC0 field descriptions (continued)

Bits	Name	Description
4 (36)	UDI	Use Debug Interrupt **NOTE:** This bit is ignored when EDBCR0EDM=1 and Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit. **NOTE:** If UDI is set and DBCR0IDM=0, no software performance monitor debug interrupts will be taken. 0 Performance Monitor Interrupts use the SRR0/1 registers and the Performance Monitor interrupt vector offset, or if the Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit, can set the EDBSR0PMI to conditionally enter debug halted mode 1 Performance Monitor Interrupts set the DBSRPMI status bit when DBCR0IDM=1, and can thus use the DSRR0/1 registers and the Debug interrupt vector offset when Debug Interrupts are not masked, or if EDBCR0EDM=1 and the Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit, can set the EDBSR0PMI to conditionally enter debug halted mode.
5:14 (37:46)	-	Reserved, should be cleared.
15:17 (47:49)	TRIGONCNTL	Trigger-on Control Class - Class of Trigger-on source 000 Trigger-on control is disabled if TRIGONSEL is 0000 (i.e. counting is not affected by triggers). All other values for TRIGONSEL are reserved. 001 Reserved 010 Trigger-on based on selected processor event(s) 011 Trigger-on based on selected hardware signal(s) 100 Trigger-on based on selected watchpoint occurrence (watchpoint #0-15) 101 Trigger-on based on selected watchpoint occurrence (extension for watchpoint #16-31) 11*x* Reserved. Indicates the condition under which triggering to start counting occurs. No triggering will occur while PMGC0FAC or PMLCanFC is set to '1'.
18 (50)	-	Reserved, should be cleared.
19:21 (51:53)	TRIGOFFCNTL	Trigger-off Control Class - Class of Trigger-off source. Indicates the condition under which triggering to stop counting occurs. No triggering will occur while PMGC0FAC or PMLCanFC is set to '1'. 000 Trigger-off control is disabled if TRIGOFFSEL is 0000 (i.e. counting is not affected by triggers) All other values for TRIGOFFSEL are reserved. 001 Reserved 010 Trigger-off based on selected processor event(s) 011 Trigger-off based on selected hardware signal(s) 100 Trigger-off based on selected watchpoint occurrence (watchpoint #0-15) 101 Trigger-off based on selected watchpoint occurrence (extension for watchpoint #16-31) 11*x* Reserved
22 (54)	-	Reserved, should be cleared.
23:26 (55:58)	TRIGONSEL	Trigger-on Source Select - Source Select based on setting of TRIGONCTL TRIGONCTL = 000:

Table continues on the next page...

Table 14-41. PMGC0 field descriptions (continued)

Bits	Name	Description
		0000 Trigger-on control is disabled 0001 - 1111 : Reserved TRIGONCTL = 010: 0000 Trigger-on when next processor interrupt occurs (software may want to set PMGC0PMIE = 0 for this setting). 0001 - 1111 : Reserved TRIGONCTL = 011: 0000 Trigger on rise of p_pmc0_qual input 0001 Trigger on rise of p_pmc1_qual input 0010 Trigger on rise of p_pmc2_qual input 0011 Trigger on rise of p_pmc3_qual input TRIGONCTL = 100: 0000 Trigger-on based on watchpoint #0 occurrence 0001 Trigger-on based on watchpoint #1 occurrence 0010 Trigger-on based on watchpoint #2 occurrence . . . 1110 Trigger-on based on watchpoint #14 occurrence 1111 Trigger-on based on watchpoint #15 occurrence TRIGONCTL = 101: 0000 Trigger-on based on watchpoint #16 occurrence 0001 Trigger-on based on watchpoint #17 occurrence 0010 Trigger-on based on watchpoint #18 occurrence . . . 1100 Trigger-on based on watchpoint #28 occurrence 1101 Trigger-on based on watchpoint #29 occurrence 1110 Trigger-on based on watchpoint #30 occurrence 1111 Trigger-on based on watchpoint #31 occurrence
27 (59)	-	Reserved, should be cleared.
28:31 (60:63)	TRIGOFFSEL	Trigger-off Source Select - Source Select based on setting of TRIGOFFCTL TRIGOFFCTL = 000: 0000 Trigger-off control is disabled 0001 - 1111 : Reserved TRIGOFFCTL = 010:

Table 14-41. PMGC0 field descriptions

Bits	Name	Description
		0000 Trigger-on when next processor interrupt occurs (software may want to set PMGC0PMIE = 0 for this setting). 0001 - 1111 : Reserved TRIGOFFCTL = 011: 0000 Trigger-off based on fall of p_pmc0_qual input 0001 Trigger-off based on fall of p_pmc1_qual input 0010 Trigger-off based on fall of p_pmc2_qual input 0011 Trigger-off based on fall of p_pmc3_qual input 0100 - 1111 : Reserved TRIGOFFCTL = 100: 0000 Trigger-off based on watchpoint #0 occurrence 0001 Trigger-off based on watchpoint #1 occurrence 0010 Trigger-off based on watchpoint #2 occurrence . . . 1110 Trigger-off based on watchpoint #14 occurrence 1111 Trigger-off based on watchpoint #15 occurrence TRIGOFFCTL = 101: 0000 Trigger-off based on watchpoint #16 occurrence 0001 Trigger-off based on watchpoint #17 occurrence 0010 Trigger-off based on watchpoint #18 occurrence . . . 1100 Trigger-off based on watchpoint #28 occurrence 1101 Trigger-off based on watchpoint #29 occurrence 1110 Trigger-off based on watchpoint #30 occurrence 1111 Trigger-off based on watchpoint #31 occurrence

14.12 Local memories

14.12.1 Local Data memory overview

Local Data (DMEM) memory has been added to provide low latency access for critical instruction routines and data operands. Access latency (zero wait-states) is similar to that of a data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity tightly coupled storage.

The local memory has one dedicated port for CPU accesses and another shared slave port for external accesses. DMEM provides a dedicated port to the CPU's load/store unit. Access via the dedicated CPU port is referred to as *front door* access. Access via the slave port is referred to as *back door* access.

Local memory occupies a portion of the processor's address space and is conditionally accessed in parallel with the corresponding cache (if present) when a CPU request is made, based on an address decode comparison early in the access pipeline.

External masters use the shared slave port, which is implemented as an AHB 2.v6 slave interface, to access the DMEM. Slave port accesses are arbitrated with the CPU port in a round-robin fashion to prevent starvation from occurring on either port.

NOTE

DMEM, by default, is always accessed through the front door (CPU) interface, regardless of whether the program uses Decorated Access, Non-decorated Cache Bypass, Atomic Load and Reserve instructions, as well as regular load/store instructions (stw, lwz, for example). The only exception is when the core's DMEMCTL0 register is set to bypass the type of DMEM access used onto AHB back door slave interface. Regular load/store instructions always access DMEM through the front door interface since no bypass control is available for these kinds of instructions.

14.12.2 Local memory control and configuration

DMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

14.12.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z4201n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

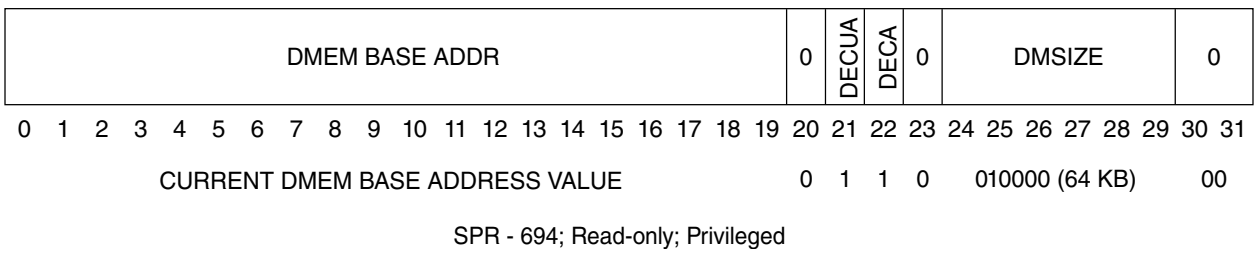


Figure 14-29. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 14-42. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

14.12.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCPECE	DSECE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18	19	20	21	22	23	24	25	26	27 28	29	30	31
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0	0	0	1	0	0	0	0	0	1 1	0	1	1

DCR - 496; Read/Write; Reset[1] - 0x0000_041B; Privileged

Figure 14-30. DMEM Control Register 0 (DMEMCTL0)

NOTE

[1] Reset by an internal power-on reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 14-43. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **NOTE:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16 :18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by

Table continues on the next page...

Table 14-43. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits.

Table continues on the next page...

Table 14-43. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written.

Table 14-43. DMEMCTL0 field descriptions

Bits	Name	Description
		1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **NOTE:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal.

14.12.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed. All other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB,\ DBYPATF,\ DBYPDEC}$.

DSWAC3	DSWAC2	DSWAC1	DSWAC0	DSRAC3	DSRAC2	DSRAC1	DSRAC0	DUWAC3	DUWAC2	DUWAC1	DUWAC0	DURAC3	DURAC2	DURAC1	DURAC0
0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31

DCR - 498; Read/Write; Reset[1] - 32'b0; Privileged

Figure 14-31. DMEM Control Register 1 (DMEMCTL1)

Note

[1] Reset from an internal power-on reset signal or by an internal reset signal.

Table 14-44. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-44. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11

Table continues on the next page...

Table 14-44. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM

Table continues on the next page...

Table 14-44. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 14-44. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

14.13 End-to-End ECC Support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

14.13.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

14.13.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

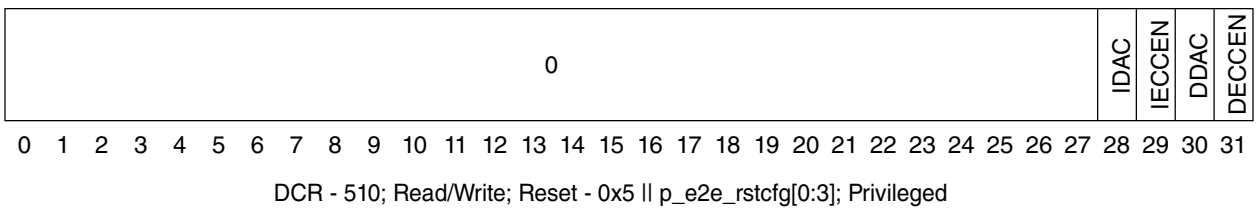


Figure 14-32. e2eECC Control Register 0 (E2ECTL0)

Table 14-45. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices.

Table 14-45. E2ECTL0 field descriptions

Bits	Name	Description
		1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

14.13.1.2 End-to-End ECC fault injection by software

Fault injection provides a way to test error recovery and portions of the error detection/ correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits). For the case of an external read access that receives an ERROR response (HRESP=ERR), the RCHKBIT field is written with all zeros.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

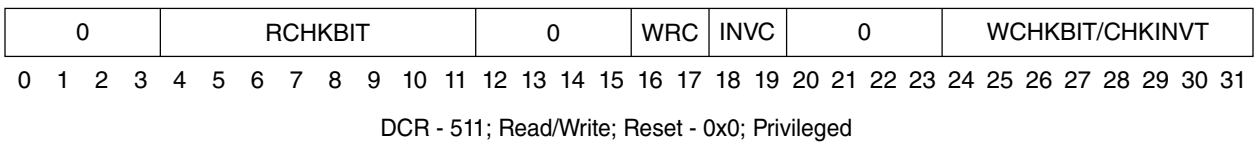


Figure 14-33. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 14-46. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked). If the CPU receives an ERROR response (HRESP=ERR) on an external read access, the RCHKBIT field is written with all zeros.
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation.

Table continues on the next page...

Table 14-46. E2EECSR0 field descriptions (continued)

Bits	Name	Description
		10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further fault injection. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 15
z7 Core Description

15.1 Overview of the e200z7260n3 core

The e200z7260n3is a dual-issue 32-bit Power ISA 2.06 Revision B VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z7260n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in Power ISA 2.06 Revision B, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The e200z7260n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z7260n3 contains a16 KB Instruction Cache, and a 16 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

15.2 Register model

This section describes the registers implemented in the e200z7260n3 core. The Power ISA 2.06 Revision B architecture defines register-to-register operations for all computational instructions.

e200z7260n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 15-1 and Figure 15-2 show the complete e200z7260n3 register set, all of which are accessible in supervisor mode. Figure 15-3 and Figure 15-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

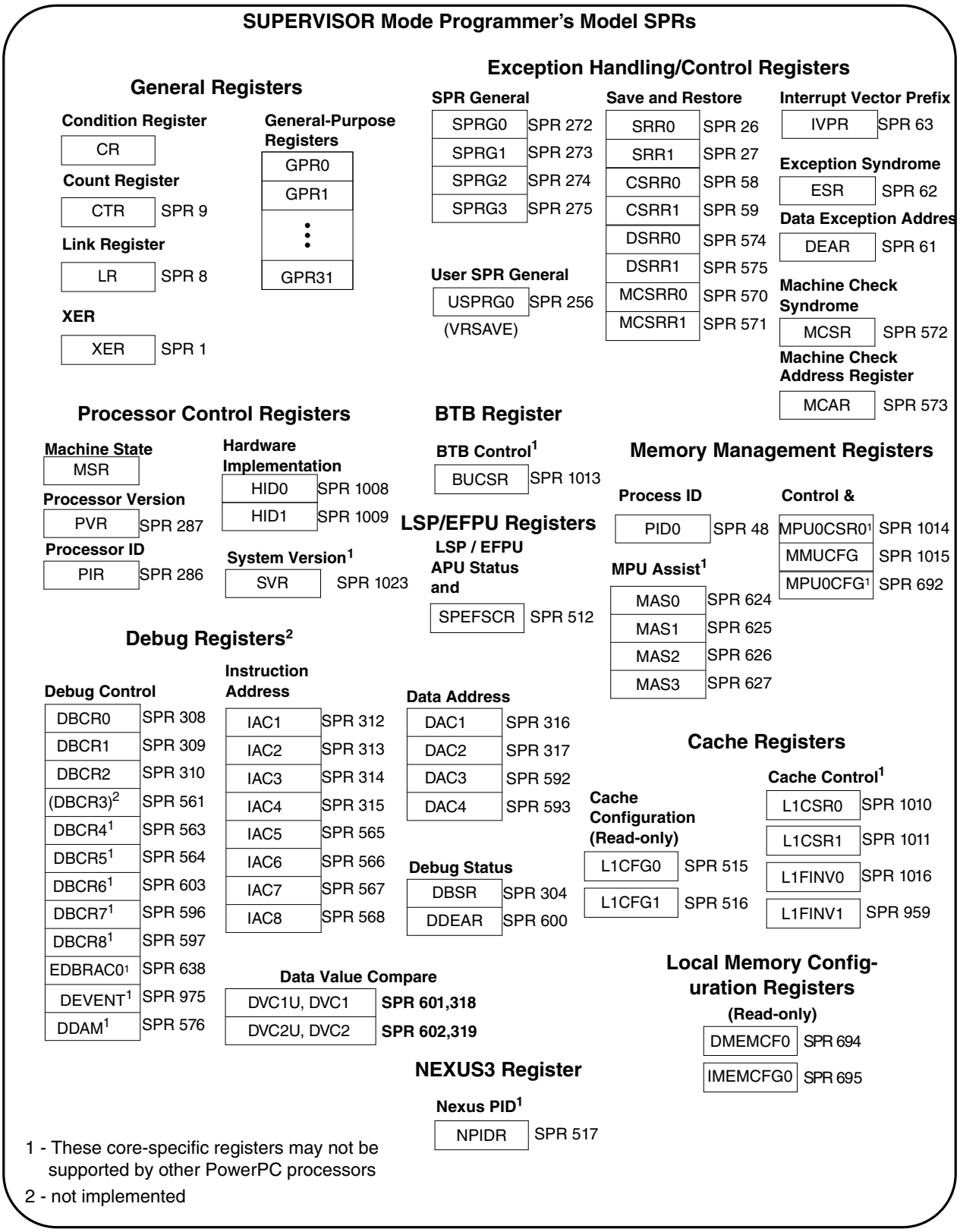


Figure 15-1. e200z7260n3 Supervisor Mode Programmer's Model SPRs

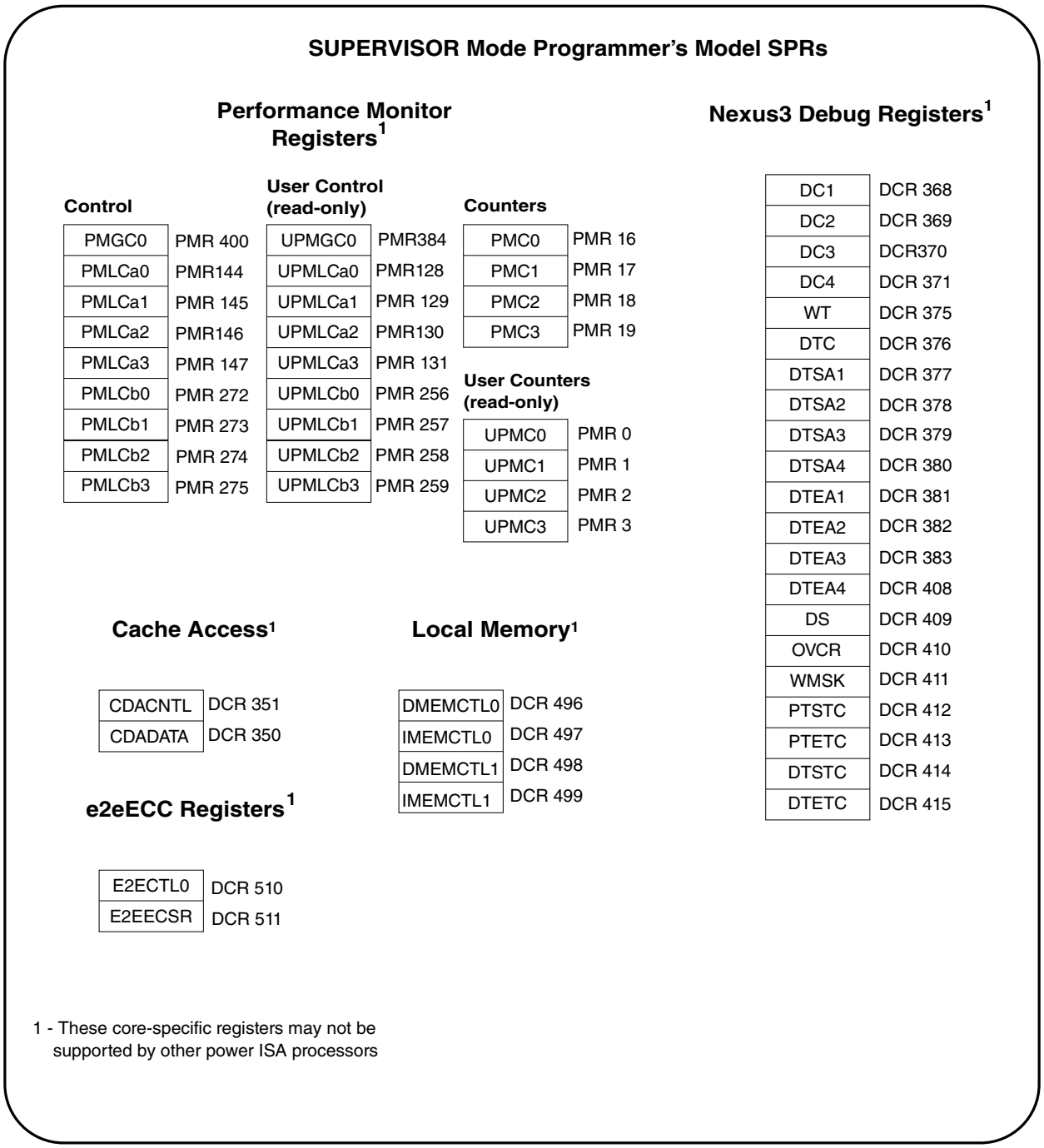


Figure 15-2. e200z7260n3 Supervisor Mode Programmer's Model DCRs and PMRs

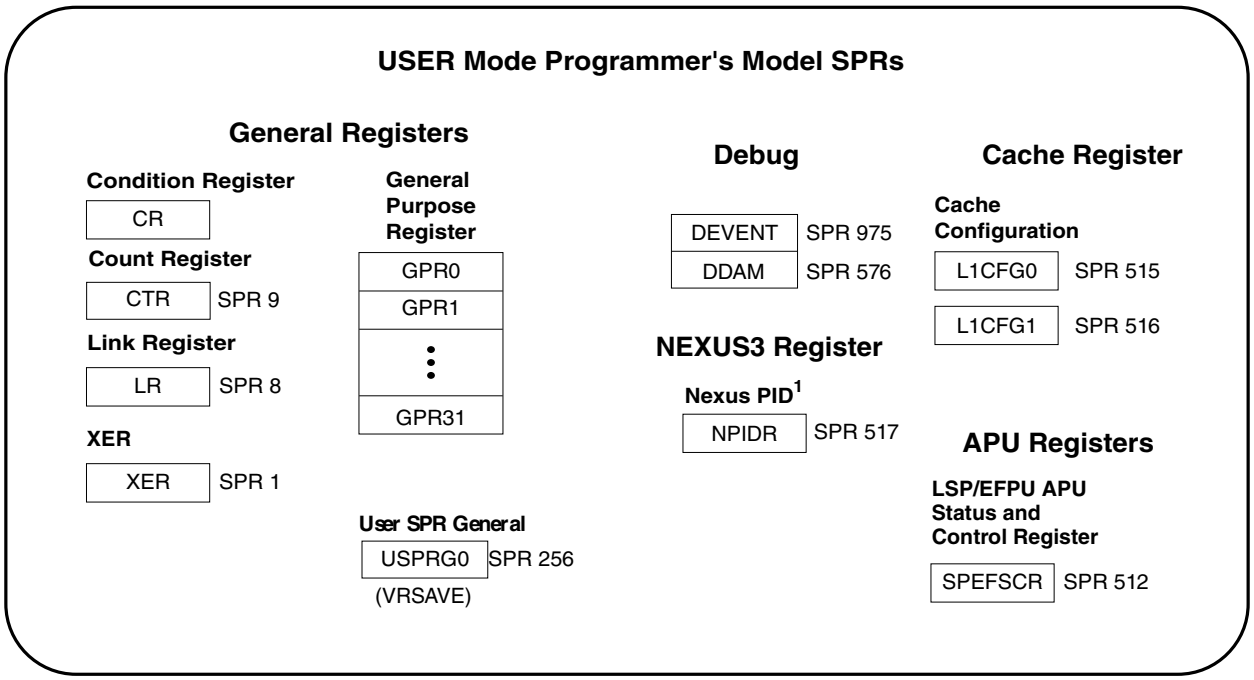


Figure 15-3. e200z7260n3 User Mode Programmer's Model SPRs

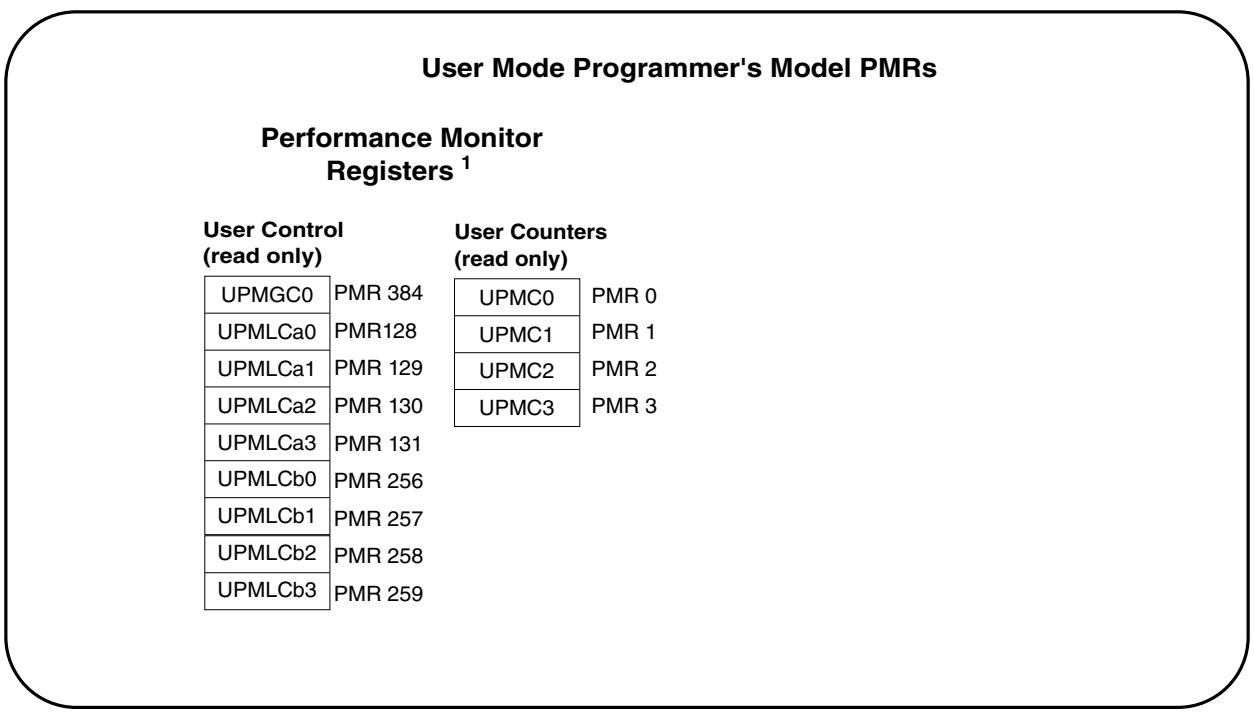
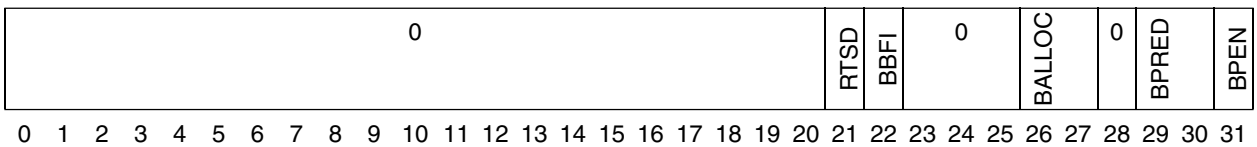


Figure 15-4. e200z7260n3 User Mode Programmer's Model PMRs

15.2.1 Branch Unit Control and Status Register (BUCSR)

The BUCSR register is used for general control and status of the branch target buffer (BTB). The BTB is detailed in Instruction prefetch buffers and branch target buffer. The BUCSR is shown in the following figure.



SPR — 1013; Read/Write; Reset — 0x0

Figure 15-5. Branch Unit Control and Status Register (BUCSR)

The BUCSR fields are defined in the following table.

Table 15-1. Branch Unit Control and Status Register (BUCSR)

Bits	Name	Description
0:20 [32:52]	-	Reserved
21 [53]	RTSD	Return Stack Disable 0 - Return Stack is enabled. 1 - Return Stack is disabled and invalidated. When written to a '1', the Return Stack is disabled, and the valid bit of all entries is cleared regardless of the value of the enable bit (BPEN). The Return Stack is only enabled when BPEN=1 and RTSD=0.
22 [54]	BBFI	Branch target buffer flash invalidate. When written to a '1', BBFI flash clears the valid bit of all entries in the branch target buffer, as well as return stack entries; clearing occurs regardless of the value of the enable bit (BPEN). Note: BBFI is always read as 0.
23:25 [55:57]	-	Reserved
26:27 [58:59]	BALLOC	Branch Target Buffer Allocation Control. This field controls BTB allocation for branch acceleration when BPEN = 1. Note that BTB hits are not affected by the settings of this field. 00 Branch Target Buffer allocation for all branches is enabled. 01 Branch Target Buffer allocation is disabled for backward branches. 10 Branch Target Buffer allocation is disabled for forward branches. 11 Branch Target Buffer allocation is disabled for both branch directions.
28 [60]	-	Reserved
29:30 [61:62]	BPRED	Branch Prediction Control (Static). This field controls operation of static prediction mechanism on a BTB miss. When a branch is predicted as taken, fetching of the predicted target location will be performed for branch acceleration. BPRED operates independently of BPEN, and with a BPEN setting of 0, will be used to perform static prediction of all unresolved branches. 00 Branch predicted taken on BTB miss for all branches. 01 Branch predicted taken on BTB miss only for forward branches.

Table continues on the next page...

Table 15-1. Branch Unit Control and Status Register (BUCSR) (continued)

Bits	Name	Description
		10 Branch predicted taken on BTB miss only for backward branches. 11 Branch predicted not taken on BTB miss for both branch directions. **NOTE:** BTB hits are not affected by the settings of this field. Note also that for certain applications, setting BPRED to a non-default value may result in improved performance.
31 [63]	BPEN	Branch target buffer prediction enable. When the BPEN bit is cleared, no hits will be generated from the BTB, and no new entries will be allocated. Entries are not automatically invalidated when BPEN is cleared; the BBFI bit controls entry invalidation. BPEN operates independently of BPRED, and will be used even with a BPRED setting of 00. 0 Branch target buffer prediction disabled 1 Branch target buffer prediction enabled (enables BTB to predict branches)

15.2.2 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a z7260n3 implementation dependent register used for various configuration and control functions. It is shown in the following figure.

SPR - 1008; Read/Write; Reset - 0x0

Figure 15-6. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 15-2. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit

Table continues on the next page...

Table 15-2. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
		1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18 [48:50]	-	Reserved
19 [51]	DCLREE	Debug Interrupt Clears MSREE 0 MSREEunaffected by Debug Interrupt 1 MSREE cleared by Debug Interrupt This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSRCE 0 MSRCE unaffected by Debug Interrupt 1 MSRCE cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSRDE 0 MSRDE unaffected by Critical class interrupt 1 MSRDE cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit. Note that if Critical Interrupt Debug events are enabled (DBCR0CIRPT set), and MSRDE is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, DSRR0DE will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSRDE enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSRDE 0 MSRDE unaffected by Machine Check interrupt 1 MSRDE cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [58:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

15.2.3 Synchronization requirements for SPRs and DCRs

With the exception of the following registers, there are no synchronization requirements for accessing SPRs and DCRs beyond those stated in PowerISA 2.06. A complete description of Synchronization requirements are contained in Chapter 12 of BookIII-E of PowerISA 2.06. Software requirements for synchronization before/after accessing these registers are shown in the following table. The notation CSI in the table refers to a Context Synchronizing instruction which include e_sc, se_sc, se_isync, se_rfi, se_rfci, se_rfmci, and se_rfdi.

Table 15-3. Additional synchronization requirements for SPRs and DCRs

Context altering event or instruction		Required before	Required after	Notes
mtmsr[PMM]		none	CSI	
mfspr, mfdcr				
DBSR	Debug Status register	msync	none	
E2EECSR0	End-to-End ECC Error Control and Status Register 0	msync	none	
HID0	Hardware Implementation-Dependent reg 0	none	none	
HID1	Hardware Implementation-Dependent reg 1	msync	none	
L1CSR0, L1CSR1, L1CSR2	L1 cache control and status registers 0,1,2	msync	none	
L1FINV0, L1FINV1	L1 cache flush and invalidate control registers 0,1	msync	none	
MPU0CSR0	MPU control and status register 0	CSI	none	
mtspr, mtdcr				
BUCSR	Branch Unit Control and Status Register	none	CSI	
DBCR0-10	Debug Control Register 0-10	none	CSI	
DBSR	Debug Status Register	msync	none	
DMEMCTL0,1	Local Data Memory Control Register 0 and 1	msync, isync	CSI	
E2ECTL0	End-to-End ECC Control and Status Register 0	msync, isync	CSI	
E2EECSR0	End-to-End ECC Error Control and Status Register 0	msync	CSI	
HID0	Hardware Implementation-Dependent reg 0	CSI	isync	
HID1	Hardware Implementation-Dependent reg 1	msync, isync	CSI	
L1CSR0	L1 cache control and status register 0	msync, isync	CSI	
L1CSR1	L1 cache control and status register 1	msync, isync	CSI	
L1CSR2	L1 cache control and status register 2	msync, isync	CSI	
L1FINV0, L1FINV1	L1 cache flush and invalidate control registers 0,1	msync	CSI	
MASx	MPU MAS registers	none	CSI	
MPU0CSR0	MPU control and status register 0	CSI	CSI	
PID (PID0)	PID0 register	CSI	CSI	
SPEFSCR	SPEFSCR register	none	CSI1	
Notes: 1. not required for status bit clearing, required for altering exception enable or rounding mode bits				

15.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

The following table shows the e200z7260n3 concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude z7260_dual-issue of those instructions.

Table 15-4. Concurrent instruction issue capabilities for e200z425n3/e200z720n3

Class of instruction	Branch	Load/ Store	Scalar integer	Scalar float	Vector integer non-MAC	Vector integer MAC	Special
Branch	—	yes	yes	yes	yes	yes	—
Load/Store	yes	—	yes	yes	yes	yes	—
Scalar integer	yes	yes	yes[1]	yes	yes	yes[2]	—
Scalar float	yes	yes	yes	—	yes	yes	—
Vector integer non-MAC	yes	yes	yes	yes	yes	yes	—
Vector integer MAC	yes	yes	yes[2]	yes	yes	—	—
Special	—	—	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.
2. Excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions.

15.4 Instruction prefetch buffers and branch target buffer

A ten-entry instruction prefetch buffer supplies instructions into the Instruction Register (IR) for decoding. Each slot in the prefetch buffer is 32 bits wide, capable of holding a single 32-bit instruction, or a pair of 16-bit instructions.

Instruction prefetches request a 64-bit doubleword and the prefetch buffer is filled with a pair of instructions at a time, except for the case of a change of flow fetch where the target is to the second (odd) word. In that case only a 32-bit prefetch is performed to load the instruction prefetch buffer. This 32-bit fetch may be immediately followed by a 64-bit prefetch to fill Slots 0 and 1 in the event that the branch is resolved to be taken.

In normal sequential execution, instructions are loaded into the IR from prefetch buffer Slot 0 and 1, and as a pair of slots are emptied, they are refilled. Whenever a pair of slots is empty, a prefetch is initiated which fills the earliest empty slot pairs beginning with Slot 0.

If the instruction prefetch buffer empties, instruction issue stalls, and the buffer is refilled. The first returned instruction is forwarded directly to the IR. Open cycles on the memory bus are utilized to keep the buffer full when possible.

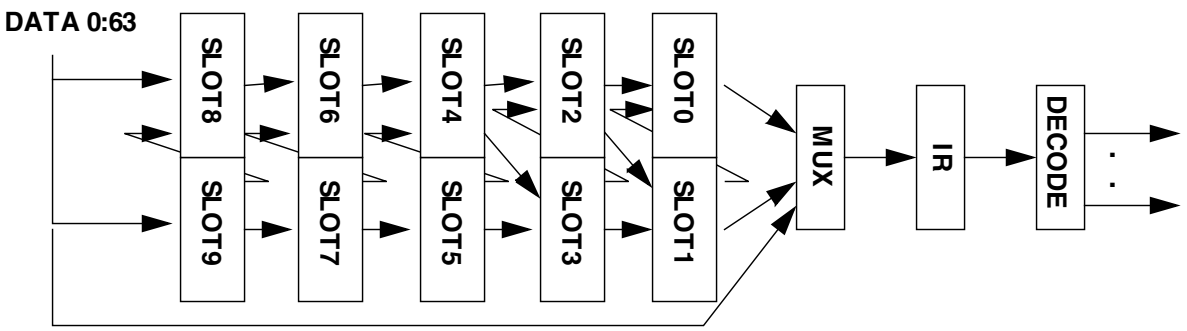


Figure 15-7. Instruction prefetch buffers

To resolve branch instructions and improve the accuracy of branch predictions, the z7260n3 implements a dynamic branch prediction mechanism using a 32/32-entry branch target buffer (BTB) and a 3-entry Return Stack used for subroutine return address prediction.

An entry is allocated in the BTB whenever a branch resolves as taken and the BTB is enabled and misses. Certain branches do not allocate BTB entries: se_bctr, se_bctrl, and se_blrl. Entries in the BTB are allocated on taken branches using a FIFO replacement algorithm.

The 32/32-entry branch target buffer structure is composed of two portions, each having 32 entries. The first portion holds non-return type branches (non-se_blr), while the second portion holds return-type instructions (se_blr) and works in conjunction with the return stack. Each BTB entry in the first portion holds a branch instruction address tag value, the branch target address, and a 2-bit branch history counter whose value is incremented or decremented on a BTB hit, depending on whether the branch was taken. Each BTB entry in the second portion holds only a branch instruction address tag value and a 2-bit branch history counter, and the branch return address is obtained from the

return stack. The counters can assume four different values: strongly taken, weakly taken, weakly not taken, and strongly not taken. On initial allocation of an entry to the BTB for a taken branch, the counter is initialized to the weakly-taken state.

The return stack is a 3-entry LIFO used to predict return addresses when a branch to link register instruction is encountered. A return stack entry is pushed whenever a branch with link instruction (e_bl, e_bcl, se_bl) is executed, or is pushed speculatively when one hits in the BTB. Return stack entries are popped when a branch to link register instruction (se_blr) is encountered. Return address predictions are validated with the current value of the link register when the se_blr instruction reaches the decode pipeline stage, and mispredictions result in a re-fetch of the correct target instruction stream.

A branch will be predicted as taken on a hit in the BTB with a counter value of strongly or weakly taken. In this case the target address contained in the BTB or return stack is used to redirect the instruction fetch stream to the target of the branch prior to the branch reaching the instruction decode stage. In the case of a BTB miss, static prediction is used to predict the outcome of the branch. In the case of a mispredicted branch, the instruction fetch stream will return to the proper instruction stream after the branch has been resolved.

When a branch is predicted taken and the branch is later resolved (in the branch execute stage), the value of the appropriate BTB counter is updated. If a branch whose counter indicates weakly taken is resolved as taken, the counter increments so that the prediction becomes strongly taken. If the branch resolves as not taken, the prediction changes to weakly not-taken. The counter saturates in the strongly taken states when the prediction is correct.

The z7260n3 does not implement the static branch prediction that is defined by the PowerISA architecture. The BO prediction information in branch encodings is ignored.

Dynamic branch prediction is enabled by setting $BUCSR_{BPEN}$. Allocation of branch target buffer entries may be controlled using the $BUCSR_{BALLOC}$ field to control whether forward or backward branches (or both) are candidates for entry into the BTB, and thus for branch prediction. Once a branch is in the BTB, $BUCSR_{ALLOC}$ has no further effect on that branch entry. Clearing $BUCSR_{BPEN}$ disables dynamic branch prediction, in which case the z7260n3 reverts to a static prediction mechanism using the $BUCSR_{BPRED}$ field to control whether forward or backward branches (or both) are predicted taken or not taken.

The BTB uses instruction fetch addresses for performing tag comparisons. On allocation of a BTB entry, the effective address of a taken branch is loaded into the entry as the tag value, and the counter value is set to weakly taken. The current PID value is not maintained as part of the tag information.

The z7260n3 does support automatic flushing of the BTB when the current PID value is updated by a mtcr PID0 instruction. Software is otherwise responsible for maintaining coherency in the BTB when a change in target address mapping is changed. This is supported by the $BUCSR_{BBFI}$ control bit.

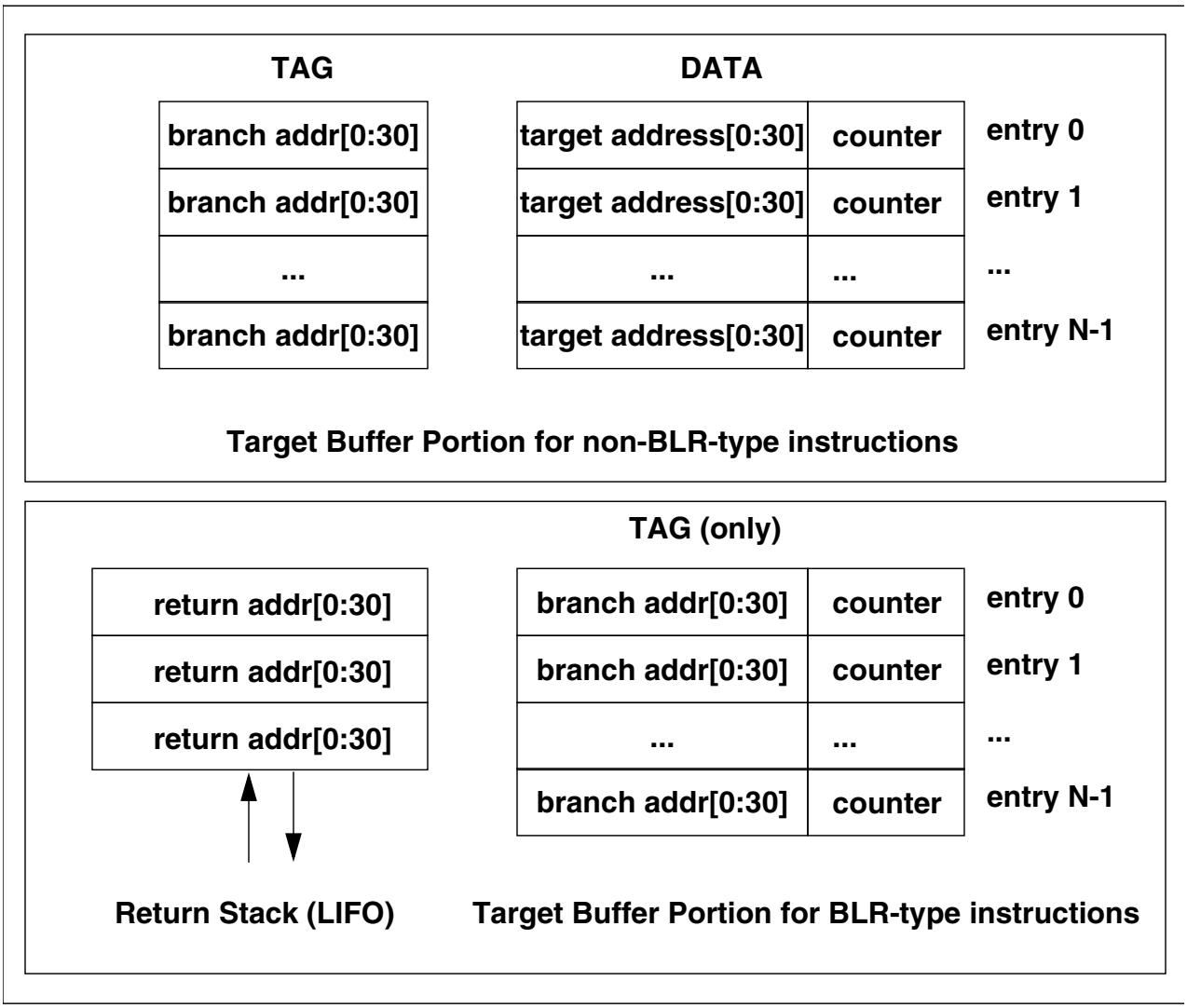


Figure 15-8. Branch target buffer with return stack

15.5 Reservation instructions and cache interactions

The CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

15.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.

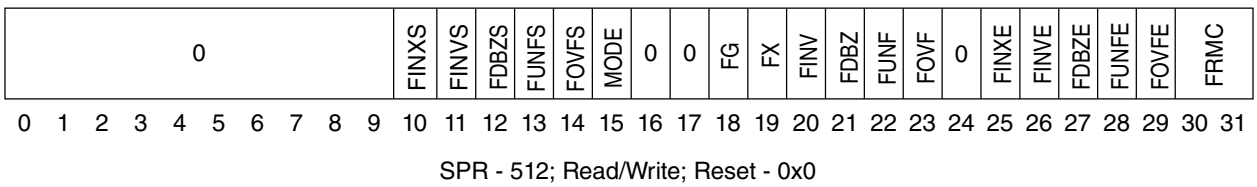


Figure 15-9. EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 15-5. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z7260n3 supports only mode 0.

Table continues on the next page...

Table 15-5. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by Zen)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled

Table continues on the next page...

Table 15-5. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

15.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.7.1 Cache overview

The e200z7260n3 processor supports a 16 KB 2-way set-associative instruction and 16KB 2-way set-associative data cache with a 32-byte line size. The caches improves system performance by providing low-latency data to the e200z7260n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z7260n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

15.7.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the a data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in the following figure.

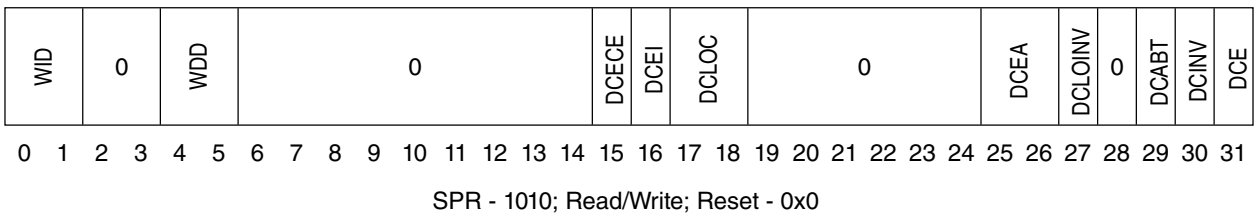


Figure 15-10. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 15-6. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved[1]
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
8:11	—	Reserved[2]
12	DCWA	Data Cache Write Allocation Policy

Table continues on the next page...

Table 15-6. L1CSR0 field descriptions (continued)

Bits	Name	Description
		This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved[2]
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18	DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. **Note:** When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[2]
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception.

Table continues on the next page...

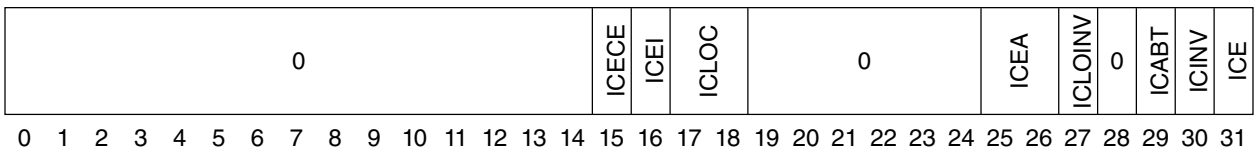
Table 15-6. L1CSR0 field descriptions (continued)

Bits	Name	Description
		01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[2]
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses.Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.7.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in the following figure.



SPR - 1011; Read/Write; Reset - 0x0

Figure 15-11. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 15-7. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved[1]
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits (p_chk[0:1]).
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one.

Table continues on the next page...

Table 15-7. L1CSR1 field descriptions (continued)

Bits	Name	Description
		11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 258 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses.

Table 15-7. L1CSR1 field descriptions

Bits	Name	Description
		Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.7.4 L1 Cache Control and Status Register 2 (L1CSR2)

The L1 Cache Control and Status Register 2 (L1CSR2) is a 32-bit register used for extended cache control. The L1CSR2 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR2 is 606 in decimal, and is shown in the following figure.

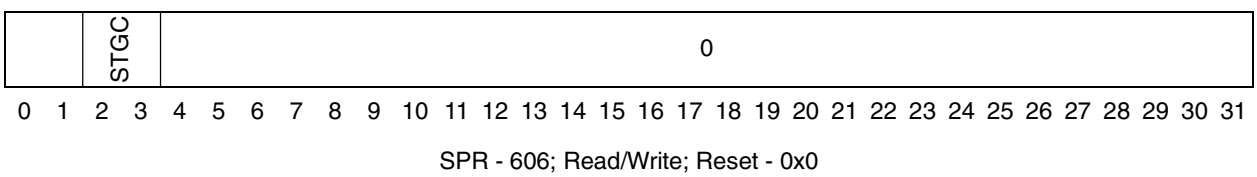


Figure 15-12. L1 Cache Control and Status Register 2 (L1CSR2)

The L1CSR2 bits are described in the following table.

Table 15-8. L1CSR2 field descriptions

Bits	Name	Description
0:1	—	Reserved
2:3	STGC	Store Gather Control 00 - Default Operation - Implementation defined. for e200 store gathering is disabled. 01 - Store Gathering is enabled. No constraints on alignment or requirement to be contiguous. External bus write accesses for gathered stores may have non-contiguous byte strobes asserted. 10 - Store Gathering is enabled. Gathered stores are required to be contiguous once gathered. Cache hit data from a store access lookup may be used to provide adjacent store data to increase gathering opportunities. External bus write accesses for gathered stores will not have non-contiguous byte strobes asserted. 11 - No Store Gathering is Performed
4:31	—	Reserved

15.7.5 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z7260n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in the following figure.

Figure 15-13. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in the following table.

Table 15-9. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x010 The size of the data cache is 10 KB

15.7.6 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z7260n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in the following figure.

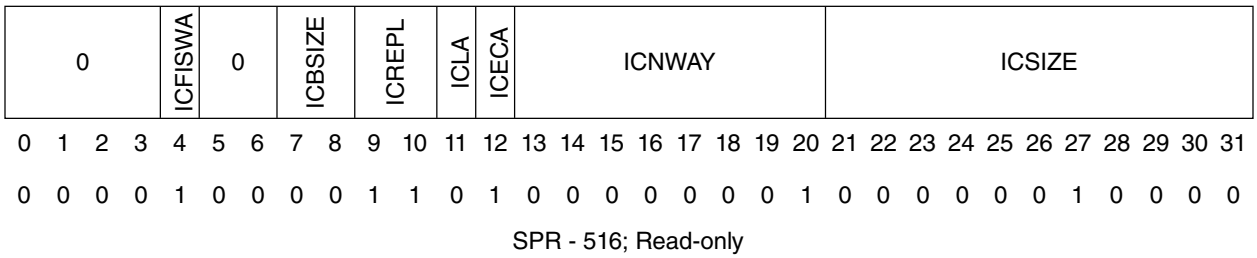


Figure 15-14. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 15-10. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x010 The size of the instruction cache is 16 KB

15.7.7 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

15.7.8 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

15.7.9 Cache Invalidate by Set and Way

e200z7260n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

15.7.9.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in the following figure.

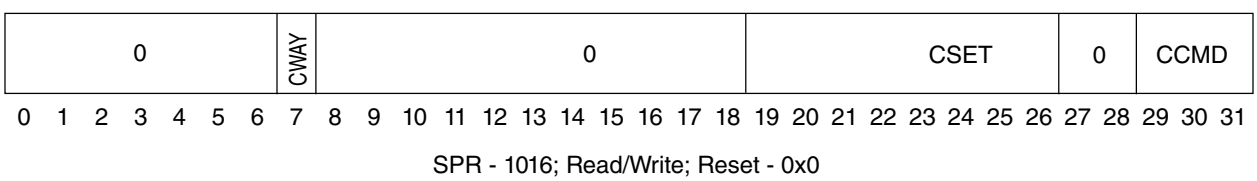


Figure 15-15. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in the following table.

Table 15-11. L1FINV0 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8: 18	—	Reserved[1] for set extension
19:26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 = The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.9.2 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in following figure.

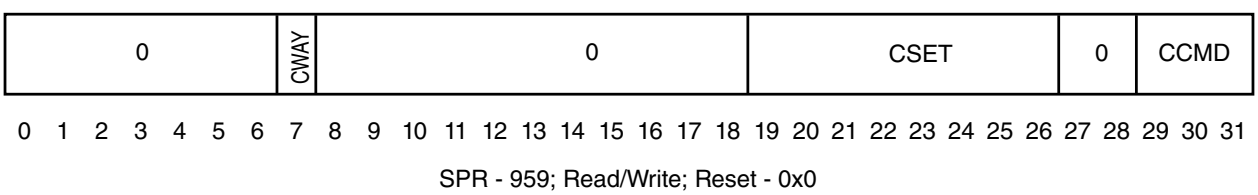


Figure 15-16. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 15-12. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved[1] for way extension

Table continues on the next page...

Table 15-12. L1FINV1 field descriptions (continued)

Bits	Name	Description
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:18	—	Reserved[1] for set extension
19 :26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.10 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the $L1CSR0_{DCECE}$and $L1CSR1_{ICECE}$ control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache ($L1CSR\{0,1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

15.7.10.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are

ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via L1CSR{0,1}$_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

15.7.11 Cache Fault Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1$_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If $L1CSR0_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction, or an L1FINV{0,1} invalidation operation.

15.8 Exceptions

Interrupts implemented in e200z7260n3 and the exception conditions that cause them are listed in the following table.

Table 15-13. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0

Table continues on the next page...

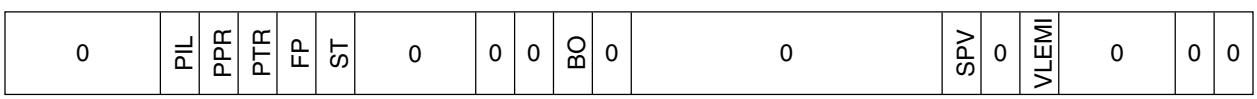
Table 15-13. Exceptions and Conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1, MPU
EFPU Data Exception	0xA0	See core specification about Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	See core specification about Embedded Floating-point Round Exception
SPE/EFPU Unavailable Exception	0xC0	See core specification about SPE/EFPU Unavailable Exception
TBD	0xD0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

15.8.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 15-17. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 15-14. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z7260n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—

Table continues on the next page...

Table 15-14. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z7260n3 MSR is shown in the following figure.

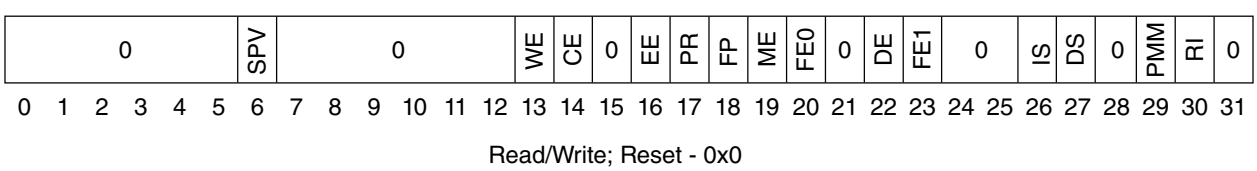


Figure 15-18. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 15-15. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/Embedded FP Double or Vector Unit instructions. **NOTE:** The e200z7260n3 processor does not support these units in hardware. An Illegal Instruction exception is generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** this bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** The e200z7260n3 processor does not support the PowerISA 2.06 floating point unit in hardware, and an Illegal Instruction exception is generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used) **NOTE:** The e200z7260n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable

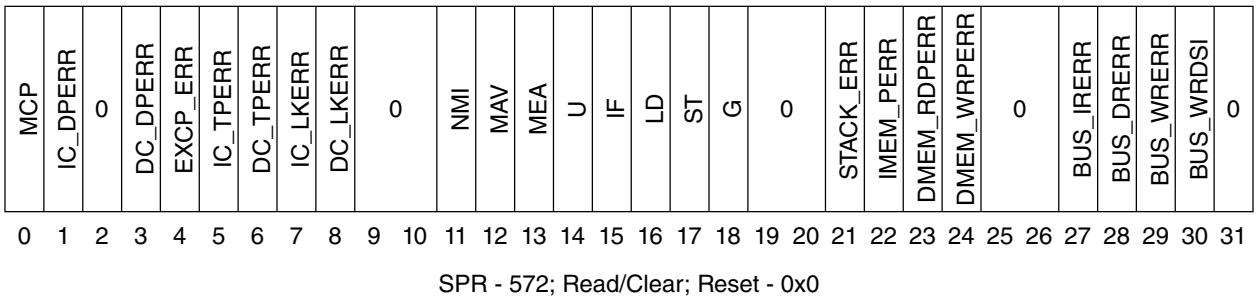
Table continues on the next page...

Table 15-15. MSR field descriptions (continued)

Bit(s)	Name	Description
		0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used) **NOTE:** The e200z7260n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** The e200z7260n3 processor does not support Address Spaces, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **NOTE:** The e200z7260n3 processor does not support Address Spaces, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

15.8.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.



SPR - 572; Read/Clear; Reset - 0x0

Figure 15-19. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 15-16. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid	Status	—

Table continues on the next page...

Table 15-16. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition		
13	MEA	MCAR holds Effective Address If MAV = 1,MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[2]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR3	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error	Async Mchk	Maybe

Table continues on the next page...

Table 15-16. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates an uncorrectable error in the DMEM on a CPU port write access.		
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
2. Note that since no address translation mechanism is present on Zen Z420n3, the MEA value is always set to \0'.
3. Will not be set by e200z7260n3 since no local instruction memory is present.

15.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

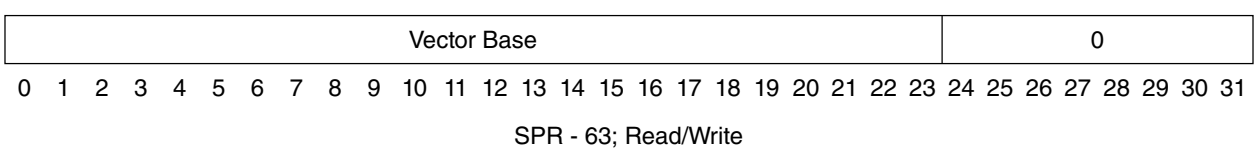


Figure 15-20. Zen Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 15-17. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.8.5 Interrupt Definitions

15.8.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 15-18. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

15.8.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 15-19. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

15.8.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 15-20. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

15.8.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 15-21. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

15.8.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 15-22. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

15.8.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

The following table lists register settings when an alignment interrupt is taken.

Table 15-23. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

15.8.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z7260n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z7260n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z7260n3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any Power ISA 2.06 Revision B floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 15-24. Program Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

15.8.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 15-25. Performance Monitor Interrupt—Register Settings

Register	Setting description					
SRR0/ DSRR0[1]	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

15.8.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 15-26. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—

Table continues on the next page...

Table 15-26. System Call Interrupt—register settings (continued)

Register	Setting description
ESR	VLEMI. All other bits cleared.
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x80

15.8.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM} = 1$ (internal debug mode) and $MSR_{DE} = 1$).

The following table lists register settings when a Debug interrupt is taken.

Table 15-27. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			

Table continues on the next page...

Table 15-27. Debug Interrupt—register settings (continued)

Register	Setting description	
	External Debug Event:	{DEVT1, DEVT2}
	Performance Monitor Debug Event:	PMI
	MPU Debug Event:	MPU
	and optionally, an	
	Imprecise Debug Event flag	{IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.8.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 15-28. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

15.8.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 15-29. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

15.8.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 15-30. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0

Table continues on the next page...

Table 15-30. System Reset Interrupt—register settings (continued)

Register	Setting description					
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

15.9 Memory Protection Unit (MPU)

15.9.1 MPU Overview

The e200z7260n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

15.9.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpu**sy**n**c instructions are privileged.

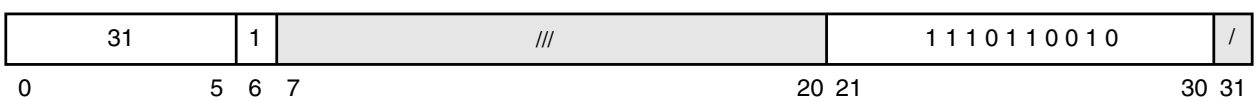
15.9.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

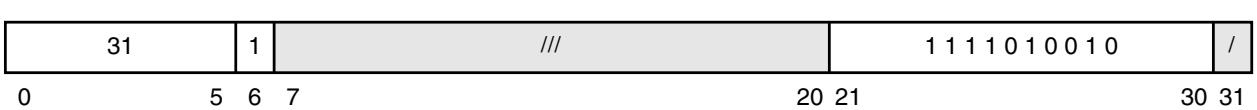
15.9.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe



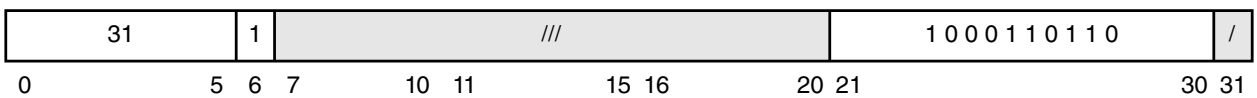
```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

15.9.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the e200z7260n3.

mpusync mpusync

MPU Synchronize
mpusync



15.9.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z7260n3 MPU design. The MMUCFG register is shown in the following figure.

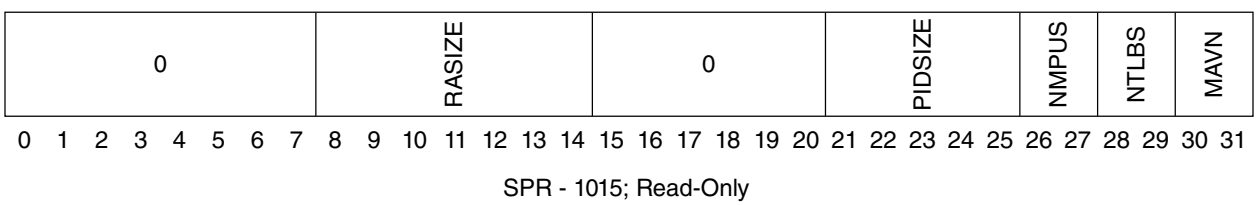


Figure 15-21. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 15-31. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU

Table continues on the next page...

Table 15-31. MMUCFG field descriptions (continued)

Bits	Name	Function
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

15.9.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z7260n3 MPU. The MPU0CFG register is shown in following figure.

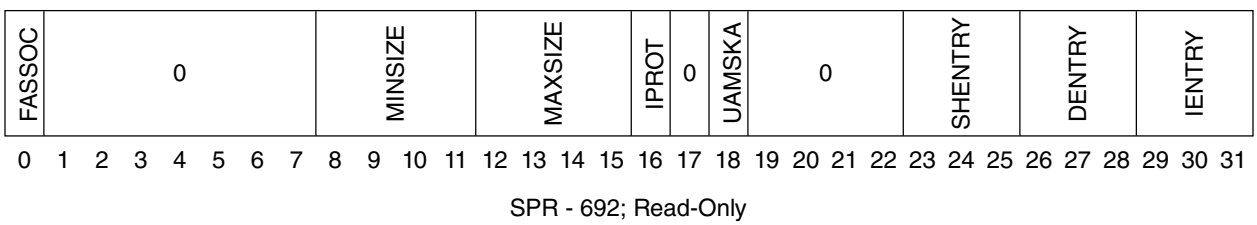


Figure 15-22. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in the following table.

Table 15-32. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved

Table continues on the next page...

Table 15-32. MPU0CFG field descriptions (continued)

Bits	Name	Function
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

15.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.

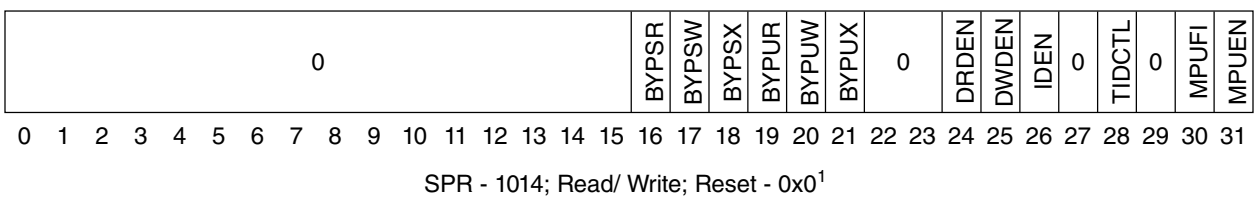


Figure 15-23. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM} = 0$, as well as unconditionally by **m_por**. If $EDBCR0_{EDM} = 1$, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 15-33. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses

Table continues on the next page...

Table 15-33. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable

Table continues on the next page...

Table 15-33. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
		This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:)** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable

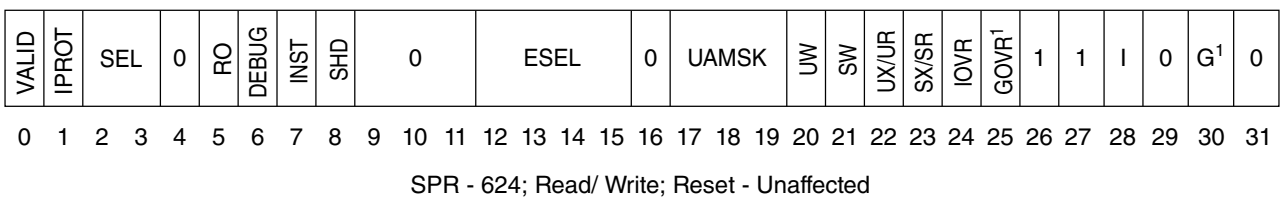
Table 15-33. MPU0CSR0 field descriptions

Bits	Name	Description
		This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

15.9.9 MPU Assist Registers (MAS)

The e200z7260n3 uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 15-24. Fields are defined in Table 15-34.



SPR - 624; Read/ Write; Reset - Unaffected

Figure 15-24. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Note

[2] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 15-34. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 = This MPU entry is invalid 1 = This MPU entry is valid
1	IPROT	Invalidation Protect 0 = Entry is not protected from invalidation

Table continues on the next page...

Table 15-34. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		1 = Entry is protected from invalidation
2:3	SEL	Selects MPU for access: 00 = Reserved, no access 01 = Reserved, no access 10 = Select MPU 11 = Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 = This MPU entry is writable 1 = This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 = This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 = This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 = This MPU entry is used for matching data accesses only 1 = This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 = The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 = The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12 :15	ESEL	Entry select for MPU.

Table continues on the next page...

Table 15-34. MAS0—MPU Read/Write and Replacement Control (continued)

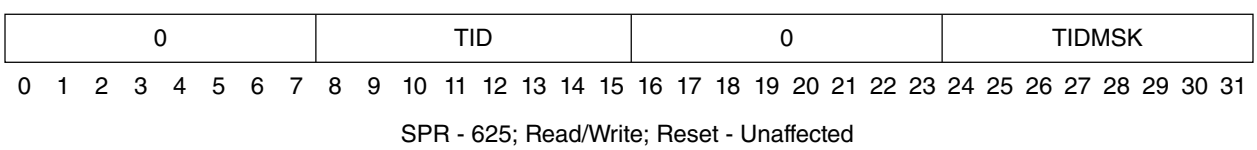
Bit	Name	Comments, or Function when Set
		This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 = No upper address bits are masked, address matching uses all 32 address bits for accesses 001 = The most significant access address bit is forced to zero before matching 010 = The two most significant access address bits are forced to zero before matching 011 = The three most significant access address bits are forced to 0 before matching 100 = The four most significant access address bits are forced to 0 before matching 101 = The upper five access address bits are forced to zero before matching 11x = Reserved, do not use
20	UW	User Mode Write PermissionDetermines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No User mode Write permission 1 = User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No Supervisor mode Write permission 1 = Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 = No User mode Execute/ Read permission 1 = User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 = No Supervisor mode Execute/ Read permission 1 = Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute OverrideDetermines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 = No override of I attribute by entry 1 = Entry I bit overrides I attribute
25	GOVR/—	G attribute Override

Table continues on the next page...

Table 15-34. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = No override of G attribute by entry 1 = Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. 0 = This region is considered cacheable 1 = This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings.This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 = All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in Figure 15-25. Fields are defined in Table 15-35.



SPR - 625; Read/Write; Reset - Unaffected

Figure 15-25. MPU Assist Register 1 (MAS1)

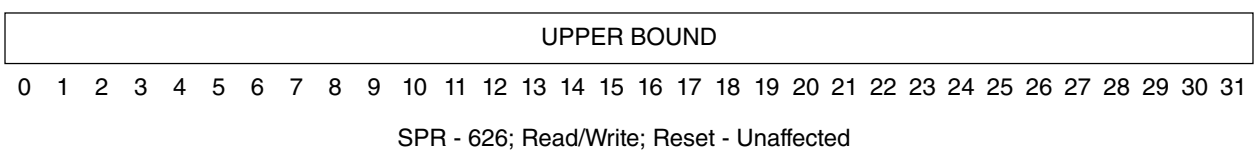
Table 15-35. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked

Table 15-35. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
		x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked xx0xxxxx = TID[2] comparison to PID0[2] not masked xx1xxxxx = TID[2] comparison to PID0[2] is masked xxx0xxxx = TID[3] comparison to PID0[3] not masked xxx1xxxx = TID[3] comparison to PID0[3] is masked xxxx0xxx = TID[4] comparison to PID0[4] not masked xxxx1xxx = TID[4] comparison to PID0[4] is masked xxxxx0xx = TID[5] comparison to PID0[5] not masked xxxxx1xx = TID[5] comparison to PID0[5] is masked xxxxxx0x = TID[6] comparison to PID0[6] not masked xxxxxx1x = TID[6] comparison to PID0[6] is masked xxxxxxx0 = TID[7] comparison to PID0[7] not masked xxxxxxx1 = TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 15-26. Fields are defined in Table 15-36.



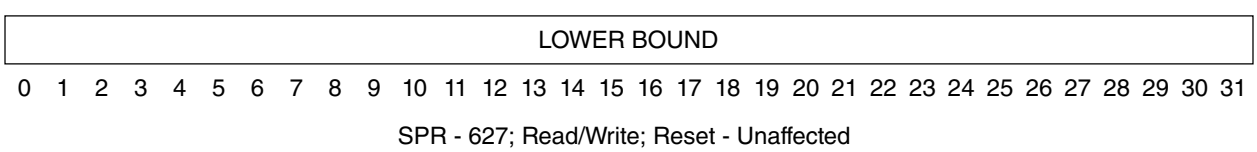
SPR - 626; Read/Write; Reset - Unaffected

Figure 15-26. MPU Assist Register 2 (MAS2)

Table 15-36. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 15-27. Fields are defined in Table 15-37.



SPR - 627; Read/Write; Reset - Unaffected

Figure 15-27. MPU Assist Register 3 (MAS3)

Table 15-37. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

15.9.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

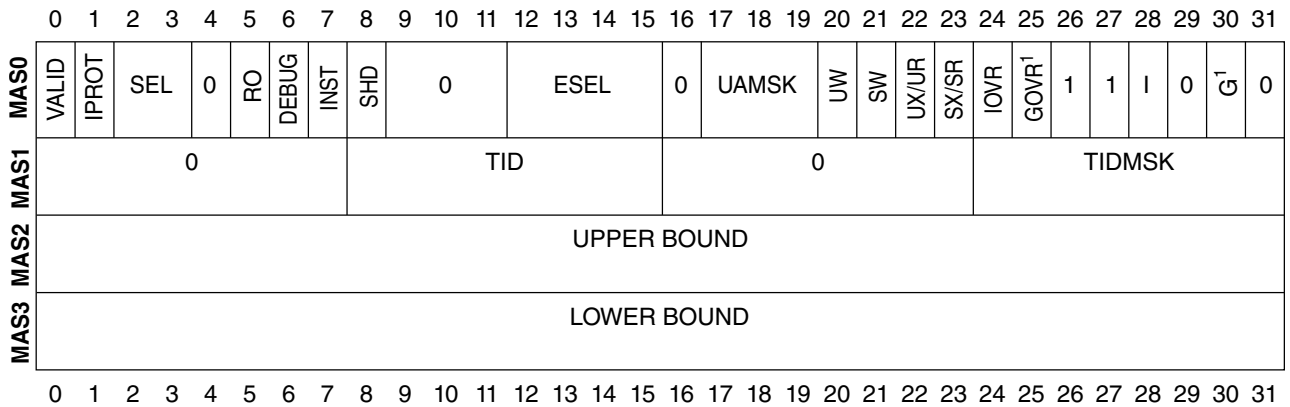


Figure 15-28. MPU Assist Registers Summary

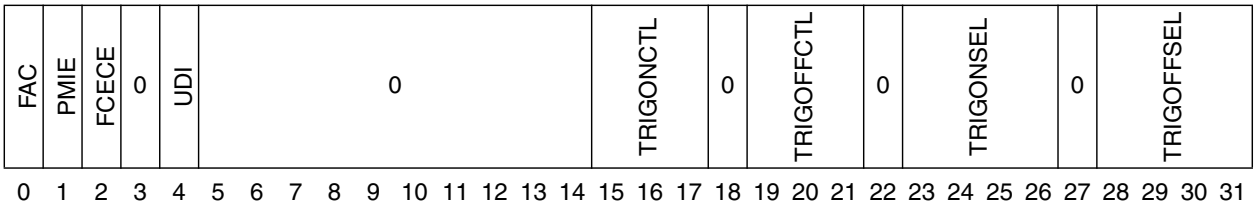
Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

15.10 Performance Monitor

15.10.1 Global Control Register 0 (PMGC0)

The performance monitor global control register PMGC0 shown in the following figure controls all performance monitor counters.



PMR - 400; Read/Write; Reset[1] - 0x0

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 15-29. Performance Monitor Global Control Register (PMGC0)

The following table describes PMGC0 fields.

Table 15-38. PMGC0 field descriptions

Bits	Name	Description
0 (32)	FAC	Freeze All Counters. 0 The PMCs are incremented (if permitted by other PMGC/PMLC control bits). 1 The PMCs are not incremented. When FAC is set to 1 by hardware or software, it has no effect on PMLCaxFC; PMLCaxFC maintains its current value until changed by software. FAC setting by hardware is controlled by PMGC0FCECE.
1 (33)	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs, at which time PMGC0PMIE is cleared Software can clear PMIE to prevent performance monitor interrupts. Performance monitor interrupts are caused by PMCx counter overflows.
2 (34)	FCECE	Freeze Counters on Enabled Condition or Event An enabled condition or event is defined as one of the following: • When the msb = 1 in PMCx and PMLCaxCE = 1. 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0FAC is set to 1. It is up to software to clear PMGC0FAC to 0.
3 (35)	-	Reserved, should be cleared.
4 (36)	UDI	Use Debug Interrupt **NOTE:** This bit is ignored when EDBCR0EDM=1 and Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit. **NOTE:** If UDI is set and DBCR0IDM=0, no software performance monitor debug interrupts will be taken.

Table continues on the next page...

Table 15-38. PMGC0 field descriptions (continued)

Bits	Name	Description
		0 Performance Monitor Interrupts use the SRR0/1 registers and the Performance Monitor interrupt vector offset, or if the Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit, can set the EDBSR0PMI to conditionally enter debug halted mode 1 Performance Monitor Interrupts set the DBSRPMI status bit when DBCR0IDM=1, and can thus use the DSRR0/1 registers and the Debug interrupt vector offset when Debug Interrupts are not masked, or if EDBCR0EDM=1 and the Performance Monitor Resources are assigned to an external debugger via the DBECR0PMI control bit, can set the EDBSR0PMI to conditionally enter debug halted mode.
5:14 (37:46)	-	Reserved, should be cleared.
15:17 (47:49)	TRIGONCNTL	Trigger-on Control Class - Class of Trigger-on source 000 Trigger-on control is disabled if TRIGONSEL is 0000 (i.e. counting is not affected by triggers). All other values for TRIGONSEL are reserved. 001 Reserved 010 Trigger-on based on selected processor event(s) 011 Trigger-on based on selected hardware signal(s) 100 Trigger-on based on selected watchpoint occurrence (watchpoint #0-15) 101 Trigger-on based on selected watchpoint occurrence (extension for watchpoint #16-31) 11*x* Reserved. Indicates the condition under which triggering to start counting occurs. No triggering will occur while PMGC0FAC or PMLCanFC is set to '1'.
18 (50)	-	Reserved, should be cleared.
19:21 (51:53)	TRIGOFFCNTL	Trigger-off Control Class - Class of Trigger-off source. Indicates the condition under which triggering to stop counting occurs. No triggering will occur while PMGC0FAC or PMLCanFC is set to '1'. 000 Trigger-off control is disabled if TRIGOFFSEL is 0000 (i.e. counting is not affected by triggers) All other values for TRIGOFFSEL are reserved. 001 Reserved 010 Trigger-off based on selected processor event(s) 011 Trigger-off based on selected hardware signal(s) 100 Trigger-off based on selected watchpoint occurrence (watchpoint #0-15) 101 Trigger-off based on selected watchpoint occurrence (extension for watchpoint #16-31) 11*x* Reserved
22 (54)	-	Reserved, should be cleared.
23:26 (55:58)	TRIGONSEL	Trigger-on Source Select - Source Select based on setting of TRIGONCTL TRIGONCTL = 000: 0000 Trigger-on control is disabled 0001 - 1111 : Reserved TRIGONCTL = 010: 0000 Trigger-on when next processor interrupt occurs (software may want to set PMGC0PMIE = 0 for this setting).

Table continues on the next page...

Table 15-38. PMGC0 field descriptions (continued)

Bits	Name	Description
		0001 - 1111 : Reserved TRIGONCTL = 011: 0000 Trigger on rise of p_pmc0_qual input 0001 Trigger on rise of p_pmc1_qual input 0010 Trigger on rise of p_pmc2_qual input 0011 Trigger on rise of p_pmc3_qual input TRIGONCTL = 100: 0000 Trigger-on based on watchpoint #0 occurrence 0001 Trigger-on based on watchpoint #1 occurrence 0010 Trigger-on based on watchpoint #2 occurrence . . . 1110 Trigger-on based on watchpoint #14 occurrence 1111 Trigger-on based on watchpoint #15 occurrence TRIGONCTL = 101: 0000 Trigger-on based on watchpoint #16 occurrence 0001 Trigger-on based on watchpoint #17 occurrence 0010 Trigger-on based on watchpoint #18 occurrence . . . 1100 Trigger-on based on watchpoint #28 occurrence 1101 Trigger-on based on watchpoint #29 occurrence 1110 Trigger-on based on watchpoint #30 occurrence 1111 Trigger-on based on watchpoint #31 occurrence
27 (59)	-	Reserved, should be cleared.
28:31 (60:63)	TRIGOFFSEL	Trigger-off Source Select - Source Select based on setting of TRIGOFFCTL TRIGOFFCTL = 000: 0000 Trigger-off control is disabled 0001 - 1111 : Reserved TRIGOFFCTL = 010: 0000 Trigger-on when next processor interrupt occurs (software may want to set PMGC0PMIE = 0 for this setting). 0001 - 1111 : Reserved TRIGOFFCTL = 011: 0000 Trigger-off based on fall of p_pmc0_qual input

Table 15-38. PMGC0 field descriptions

Bits	Name	Description
		0001 Trigger-off based on fall of p_pmc1_qual input 0010 Trigger-off based on fall of p_pmc2_qual input 0011 Trigger-off based on fall of p_pmc3_qual input 0100 - 1111 : Reserved TRIGOFFCTL = 100: 0000 Trigger-off based on watchpoint #0 occurrence 0001 Trigger-off based on watchpoint #1 occurrence 0010 Trigger-off based on watchpoint #2 occurrence . . . 1110 Trigger-off based on watchpoint #14 occurrence 1111 Trigger-off based on watchpoint #15 occurrence TRIGOFFCTL = 101: 0000 Trigger-off based on watchpoint #16 occurrence 0001 Trigger-off based on watchpoint #17 occurrence 0010 Trigger-off based on watchpoint #18 occurrence . . . 1100 Trigger-off based on watchpoint #28 occurrence 1101 Trigger-off based on watchpoint #29 occurrence 1110 Trigger-off based on watchpoint #30 occurrence 1111 Trigger-off based on watchpoint #31 occurrence

15.11 Local memories

15.11.1 Local Data memory overview

Local Data (DMEM) memory has been added to provide low latency access for critical instruction routines and data operands. Access latency (zero wait-states) is similar to that of a data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity tightly coupled storage.

The local memory has one dedicated port for CPU accesses and another shared slave port for external accesses. DMEM provides a dedicated port to the CPU's load/store unit. Access via the dedicated CPU port is referred to as *front door* access. Access via the slave port is referred to as *back door* access.

Local memory occupies a portion of the processor's address space and is conditionally accessed in parallel with the corresponding cache (if present) when a CPU request is made, based on an address decode comparison early in the access pipeline.

External masters use the shared slave port, which is implemented as an AHB 2.v6 slave interface, to access the DMEM. Slave port accesses are arbitrated with the CPU port in a round-robin fashion to prevent starvation from occurring on either port.

NOTE

DMEM, by default, is always accessed through the front door (CPU) interface, regardless of whether the program uses Decorated Access, Non-decorated Cache Bypass, Atomic Load and Reserve instructions, as well as regular load/store instructions (stw, lwz, for example). The only exception is when the core's DMEMCTL0 register is set to bypass the type of DMEM access used onto AHB back door slave interface. Regular load/store instructions always access DMEM through the front door interface since no bypass control is available for these kinds of instructions.

15.11.2 Local memory control and configuration

DMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.11.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z7260n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

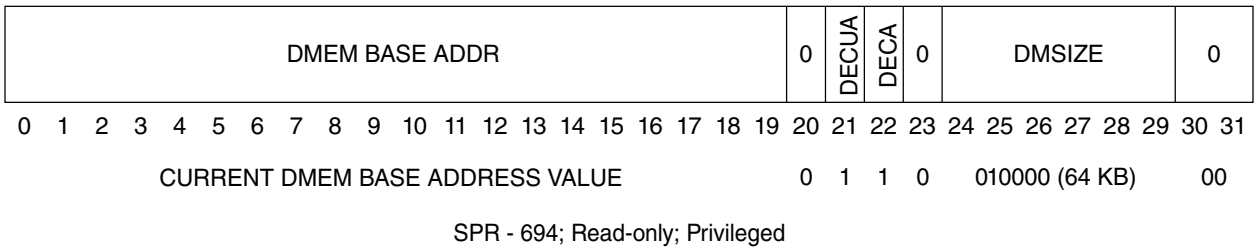


Figure 15-30. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 15-39. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

15.11.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCPECE	DSECE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1

DCR - 496; Read/Write; Reset[1] - 0x0000_041B; Privileged

Figure 15-31. DMEM Control Register 0 (DMEMCTL0)

Note

[1] Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 15-40. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16 :18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated on **m_por** and **slv_reset_b**.
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated on **m_por** and **slv_reset_b**.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses

Table continues on the next page...

Table 15-40. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated on **m_por** and **slv_reset_b**.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) **Note:** This field is updated on **m_por** and **slv_reset_b**.
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port (**p_dmem_baseaddr[0:n]**) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]).
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors.

Table continues on the next page...

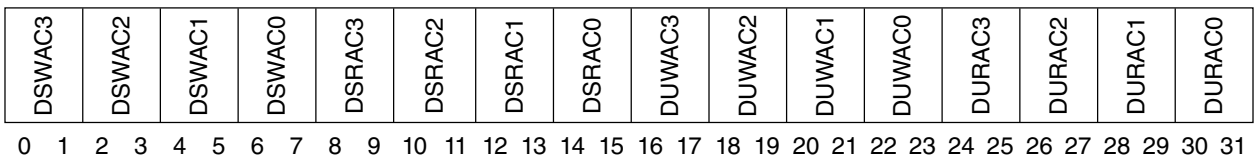
Table 15-40. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. **Note:** This field is updated on **m_por** and **slv_reset_b**.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **m_por** and **slv_reset_b**.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated on **m_por** and **slv_reset_b**.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** This field is updated on **m_por** and **slv_reset_b**.

15.11.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU

memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB, DBYPATF, DBYPDEC}$.



DCR - 498; Read/Write; Reset[1] - 32'b0; Privileged

Figure 15-32. DMEM Control Register 1 (DMEMCTL1)

Note

[1] Reset by **m_por** and **p_reset_b**.

Table 15-41. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 15-41. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01

Table continues on the next page...

Table 15-41. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM

Table continues on the next page...

Table 15-41. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved

Table 15-41. DMEMCTL1 field descriptions

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

15.12 End-to-End ECC Support

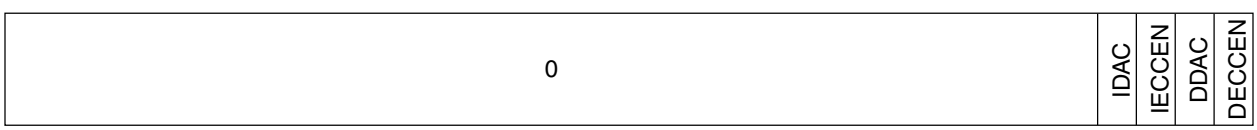
This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

15.12.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

15.12.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.



DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 15-33. e2eECC Control Register 0 (E2ECTL0)

Table 15-42. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses.

Table continues on the next page...

Table 15-42. E2ECTL0 field descriptions (continued)

Bits	Name	Description
		0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.12.1.2 End-to-End ECC fault injection by software

Fault injection provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits).

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16	17	18 19 20 21	22 23 24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 15-34. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 15-43. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 16
Nexus Module

16.1 Introduction

The e200z4201n3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

16.1.1 General description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

16.1.2 Terms and definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 16-1. Terms and definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

16.1.3 Feature list

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)
- Indicates to the FCCU whether it becomes active during functional mode.

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

16.1.4 Functional block diagram

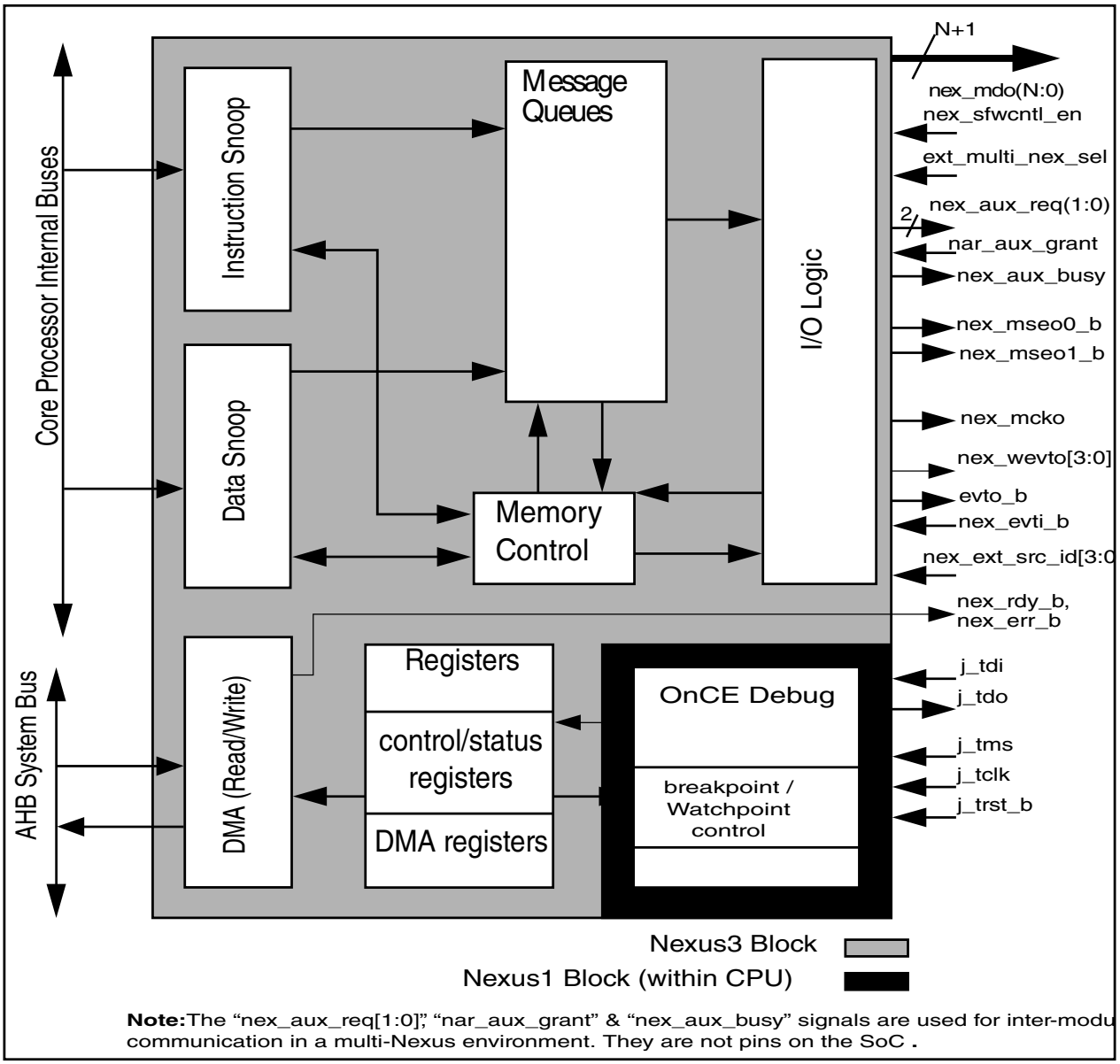


Figure 16-1. Nexus 3 functional block diagram

16.2 Enabling Nexus 3 operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

16.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

16.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 16-2.

Table 16-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 16-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table continues on the next page...

Table 16-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 16-7)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)

Table continues on the next page...

Table 16-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 16-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 16-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 16-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions

Table continues on the next page...

Table 16-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 16-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 16-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 16-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 16-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 16-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 16-5 shows the encodings used for resource codes for certain messages.

Table 16-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 16-6 shows the event code encodings used for certain messages.

Table 16-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace

Table continues on the next page...

Table 16-6. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 16-7 shows the data trace size encodings used for certain messages.

Table 16-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

16.4 Nexus 3 Programmer's model

This section describes the Nexus 3 programmer's model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 16-8. Nexus 3 register map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.

3. The "PCR_INDEX" is a parameter determined by the SoC.

16.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

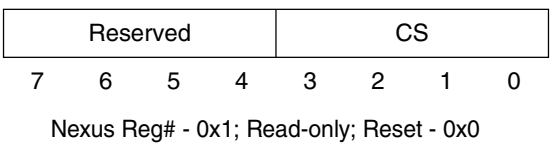


Figure 16-2. Client Select Control (CSC) register

Table 16-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

16.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

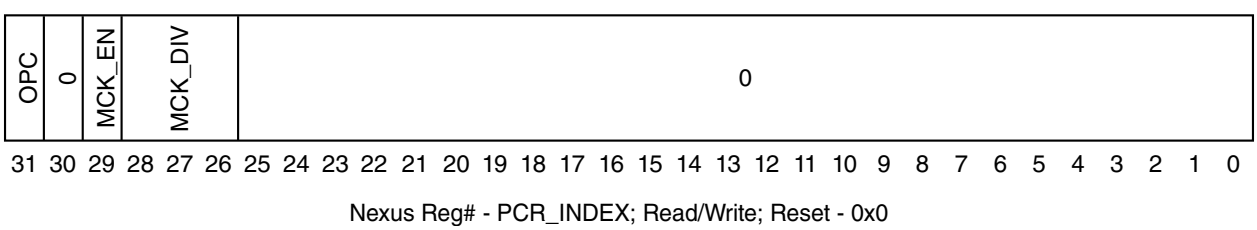


Figure 16-3. Port Configuration Register

Table 16-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality

Table continues on the next page...

Table 16-10. PCR field descriptions (continued)

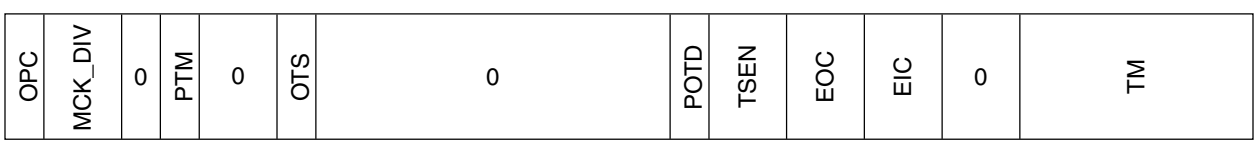
Bit	Name	Description
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

16.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 16-4 and its fields are described in Table 16-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 16-4. Development Control Register 1

Table 16-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.

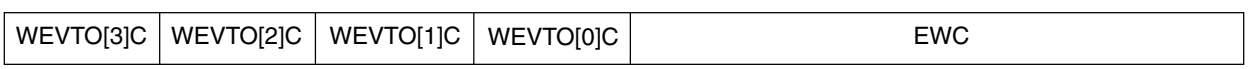
Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

16.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z4201n3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 16-5 and its fields are described in Table 16-12.



Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 16-5. Development Control Register 2

Table 16-12. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[3]** 0001 Watchpoint #0 triggers **nex_wevto[3]** 0010 Watchpoint #1 triggers **nex_wevto[3]** 0011 Watchpoint #2 triggers **nex_wevto[3]** 0100 Watchpoint #3 triggers **nex_wevto[3]**

Table continues on the next page...

Table 16-12. DC2 field descriptions (continued)

Bits	Name	Description
		0101 Watchpoint #4 triggers **nex_wevto[3]**
		0110 Watchpoint #5 triggers **nex_wevto[3]**
		0111 Watchpoint #6 triggers **nex_wevto[3]**
		1000 Watchpoint #7 triggers **nex_wevto[3]**
		1001 Watchpoint #8 triggers **nex_wevto[3]**
		1010 Watchpoint #9 triggers **nex_wevto[3]**
		1011 Watchpoint #10 triggers **nex_wevto[3]**
		1100 Watchpoint #11 triggers **nex_wevto[3]**
		1101 Watchpoint #12 triggers **nex_wevto[3]**
		1110 Watchpoint #13 triggers **nex_wevto[3]**
		1111 Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[2]**
		0001 Watchpoint #0 triggers **nex_wevto[2]**
		0010 Watchpoint #1 triggers **nex_wevto[2]**
		0011 Watchpoint #2 triggers **nex_wevto[2]**
		0100 Watchpoint #3 triggers **nex_wevto[2]**
		0101 Watchpoint #4 triggers **nex_wevto[2]**
		0110 Watchpoint #5 triggers **nex_wevto[2]**
		0111 Watchpoint #6 triggers **nex_wevto[2]**
		1000 Watchpoint #7 triggers **nex_wevto[2]**
		1001 Watchpoint #8 triggers **nex_wevto[2]**
		1010 Watchpoint #9 triggers **nex_wevto[2]**
		1011 Watchpoint #10 triggers **nex_wevto[2]**
		1100 Watchpoint #11 triggers **nex_wevto[2]**
		1101 Watchpoint #12 triggers **nex_wevto[2]**
		1110 Watchpoint #13 triggers **nex_wevto[2]**
		1111 Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[1]**
		0001 Watchpoint #0 triggers **nex_wevto[1]**
		0010 Watchpoint #1 triggers **nex_wevto[1]**
		0011 Watchpoint #2 triggers **nex_wevto[1]**
		0100 Watchpoint #3 triggers **nex_wevto[1]**
		0101 Watchpoint #4 triggers **nex_wevto[1]**
		0110 Watchpoint #5 triggers **nex_wevto[1]**
		0111 Watchpoint #6 triggers **nex_wevto[1]**

Table continues on the next page...

Table 16-12. DC2 field descriptions (continued)

Bits	Name	Description
		1000 Watchpoint #7 triggers **nex_wevto[1]**
		1001 Watchpoint #8 triggers **nex_wevto[1]**
		1010 Watchpoint #9 triggers **nex_wevto[1]**
		1011 Watchpoint #10 triggers **nex_wevto[1]**
		1100 Watchpoint #11 triggers **nex_wevto[1]**
		1101 Watchpoint #12 triggers **nex_wevto[1]**
		1110 Watchpoint #13 triggers **nex_wevto[1]**
		1111 Watchpoint #14 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[0]**
		0001 Watchpoint #0 triggers **nex_wevto[0]**
		0010 Watchpoint #1 triggers **nex_wevto[0]**
		0011 Watchpoint #2 triggers **nex_wevto[0]**
		0100 Watchpoint #3 triggers **nex_wevto[0]**
		0101 Watchpoint #4 triggers **nex_wevto[0]**
		0110 Watchpoint #5 triggers **nex_wevto[0]**
		0111 Watchpoint #6 triggers **nex_wevto[0]**
		1000 Watchpoint #7 triggers **nex_wevto[0]**
		1001 Watchpoint #8 triggers **nex_wevto[0]**
		1010 Watchpoint #9 triggers **nex_wevto[0]**
		1011 Watchpoint #10 triggers **nex_wevto[0]**
		1100 Watchpoint #11 triggers **nex_wevto[0]**
		1101 Watchpoint #12 triggers **nex_wevto[0]**
		1110 Watchpoint #13 triggers **nex_wevto[0]**
		1111 Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration[1]
		0000000000000000 No Watchpoints #0-15 trigger **evto_b**
		XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **evto_b**
		XXXXXXXXXXXXXX1X Watchpoint #1 triggers **evto_b**
		XXXXXXXXXXXXX1XX Watchpoint #2 triggers **evto_b**
		XXXXXXXXXXXX1XXX Watchpoint #3 triggers **evto_b**
		XXXXXXXXXXX1XXXX Watchpoint #4 triggers **evto_b**
		XXXXXXXXXX1XXXXX Watchpoint #5 triggers **evto_b**
		XXXXXXXXX1XXXXXX Watchpoint #6 triggers **evto_b**
		XXXXXXXX1XXXXXXX Watchpoint #7 triggers **evto_b**
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **evto_b**
		XXXXXX1XXXXXXXXX Watchpoint #9 triggers **evto_b**

Table 16-12. DC2 field descriptions

Bits	Name	Description
		XXXXX1XXXXXXXXXX Watchpoint #10 triggers **evto_b**
		XXXX1XXXXXXXXXXX Watchpoint #11 triggers **evto_b**
		XXX1XXXXXXXXXXXX Watchpoint #12 triggers **evto_b**
		XX1XXXXXXXXXXXXX Watchpoint #13 triggers **evto_b**
		X1XXXXXXXXXXXXXX Watchpoint #14 triggers **evto_b**
		1XXXXXXXXXXXXXXX Watchpoint #15 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 16-6 and its fields are described in Table 16-13.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 16-6. Development Control 3 (DCR3) register

Table 16-13. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration
		0000 No Watchpoints #15-#26 trigger **nex_wevto[3]**
		0001 Watchpoint #15 triggers **nex_wevto[3]**
		0010 Watchpoint #16 triggers **nex_wevto[3]**
		0011 Watchpoint #17 triggers **nex_wevto[3]**
		0100 Watchpoint #18 triggers **nex_wevto[3]**
		0101 Watchpoint #19 triggers **nex_wevto[3]**
		0110 Watchpoint #20 triggers **nex_wevto[3]**
		0111 Watchpoint #21 triggers **nex_wevto[3]**
		1000 Watchpoint #22 triggers **nex_wevto[3]**
		1001 Watchpoint #23 triggers **nex_wevto[3]**
		1010 Watchpoint #24 triggers **nex_wevto[3]**
		1011 Watchpoint #25 triggers **nex_wevto[3]**
		1100 Watchpoint #26 triggers **nex_wevto[3]**
		1101 - 1111 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration
		0000 No Watchpoints #15-#26 trigger **nex_wevto[2]**
		0001 Watchpoint #15 triggers **nex_wevto[2]**
		0010 Watchpoint #16 triggers **nex_wevto[2]**

Table continues on the next page...

Table 16-13. DC3 field descriptions (continued)

Bits	Name	Description
		0011 Watchpoint #17 triggers **nex_wevto[2]**
		0100 Watchpoint #18 triggers **nex_wevto[2]**
		0101 Watchpoint #19 triggers **nex_wevto[2]**
		0110 Watchpoint #20 triggers **nex_wevto[2]**
		0111 Watchpoint #21 triggers **nex_wevto[2]**
		1000 Watchpoint #22 triggers **nex_wevto[2]**
		1001 Watchpoint #23 triggers **nex_wevto[2]**
		1010 Watchpoint #24 triggers **nex_wevto[2]**
		1011 Watchpoint #25 triggers **nex_wevto[2]**
		1100 Watchpoint #26 triggers **nex_wevto[2]**
		1101 - 1111 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration
		0000 No Watchpoints #15-#29 trigger **nex_wevto[1]**
		0001 Watchpoint #15 triggers **nex_wevto[1]**
		0010 Watchpoint #16 triggers **nex_wevto[1]**
		0011 Watchpoint #17 triggers **nex_wevto[1]**
		0100 Watchpoint #18 triggers **nex_wevto[1]**
		0101 Watchpoint #19 triggers **nex_wevto[1]**
		0110 Watchpoint #20 triggers **nex_wevto[1]**
		0111 Watchpoint #21 triggers **nex_wevto[1]**
		1000 Watchpoint #22 triggers **nex_wevto[1]**
		1001 Watchpoint #23 triggers **nex_wevto[1]**
		1010 Watchpoint #24 triggers **nex_wevto[1]**
		1011 Watchpoint #25 triggers **nex_wevto[1]**
		1100 Watchpoint #26 triggers **nex_wevto[1]**
		1101 Watchpoint #27 triggers **nex_wevto[1]**
		1110 Watchpoint #28 triggers **nex_wevto[1]**
		1111 Watchpoint #29 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #15-#29 trigger **nex_wevto[0]**
		0001 Watchpoint #15 triggers **nex_wevto[0]**
		0010 Watchpoint #16 triggers **nex_wevto[0]**
		0011 Watchpoint #17 triggers **nex_wevto[0]**
		0100 Watchpoint #18 triggers **nex_wevto[0]**
		0101 Watchpoint #19 triggers **nex_wevto[0]**
		0110 Watchpoint #20 triggers **nex_wevto[0]**
		0111 Watchpoint #21 triggers **nex_wevto[0]**

Table continues on the next page...

Table 16-13. DC3 field descriptions (continued)

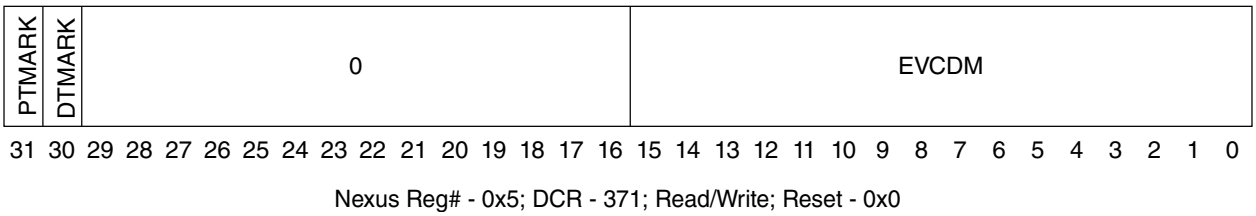
Bits	Name	Description
		1000 Watchpoint #22 triggers **nex_wevto[0]**
		1001 Watchpoint #23 triggers **nex_wevto[0]**
		1010 Watchpoint #24 triggers **nex_wevto[0]**
		1011 Watchpoint #25 triggers **nex_wevto[0]**
		1100 Watchpoint #26 triggers **nex_wevto[0]**
		1101 Watchpoint #27 triggers **nex_wevto[0]**
		1110 Watchpoint #28 triggers **nex_wevto[0]**
		1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1]
		00000000000000 No Watchpoints #16-#29 trigger **evto_b**
		XXXXXXXXXXXXX1 Watchpoint #16 triggers **evto_b**
		XXXXXXXXXXXX1X Watchpoint #17 triggers **evto_b**
		XXXXXXXXXXX1XX Watchpoint #18 triggers **evto_b**
		XXXXXXXXXX1XXX Watchpoint #19 triggers **evto_b**
		XXXXXXXXX1XXXX Watchpoint #20 triggers **evto_b**
		XXXXXXXX1XXXXX Watchpoint #21 triggers **evto_b**
		XXXXXXX1XXXXXX Watchpoint #22 triggers **evto_b**
		XXXXXX1XXXXXXX Watchpoint #23 triggers **evto_b**
		XXXXX1XXXXXXXX Watchpoint #24 triggers **evto_b**
		XXXX1XXXXXXXXX Watchpoint #25 triggers **evto_b**
		XXX1XXXXXXXXXX Watchpoint #26 triggers **evto_b**
		XX1XXXXXXXXXXX Watchpoint #27 triggers **evto_b**
		X1XXXXXXXXXXXX Watchpoint #28 triggers **evto_b**
		1XXXXXXXXXXXXX Watchpoint #29 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

16.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 16-7 and its fields are described in Table 16-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

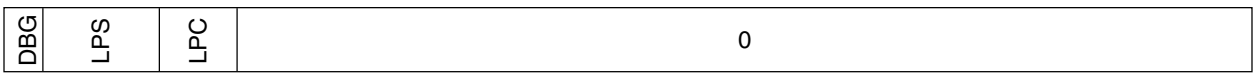
Figure 16-7. Development Control 4 (DC4) register

Table 16-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask For implemented EVCODEs. please see Table 16-6. 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

16.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 16-8. Development Status Register

Table 16-15. DS field description

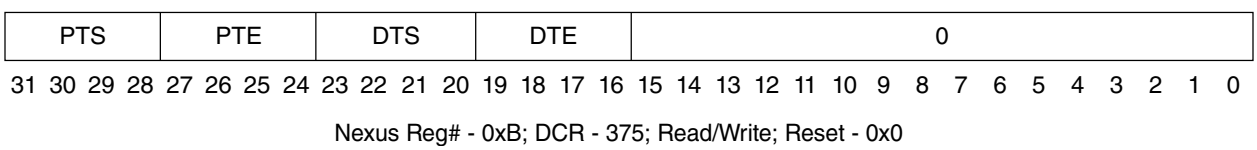
Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

16.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data

Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z4201n3) Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 16-9. Watchpoint Trigger (WT) Register

Table 16-16 details the Watchpoint Trigger register fields.

Table 16-16. WT field descriptions

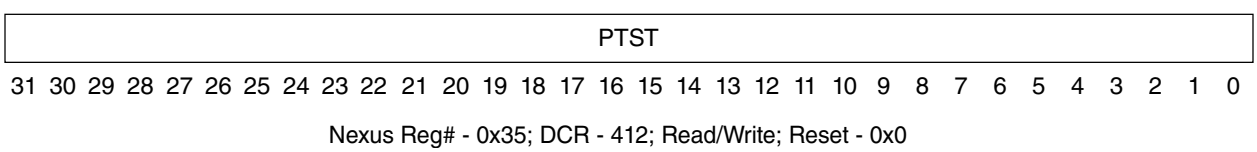
Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register

Table continues on the next page...

Table 16-16. WT field descriptions (continued)

Bits	Name	Description
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.



Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 16-10. Program Trace Start Trigger Control (PTSTC) register

Table 16-17 details the PTSTC register fields.

Table 16-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control

Table continues on the next page...

Table 16-17. Program Trace Start Trigger Control Register Fields (continued)

Bits	Name	Description
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 16-11. Program Trace End Trigger Control (PTETC) register

Table 16-18 details the PTETC register fields.

Table 16-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28

Table 16-18. PTETC field descriptions

Bits	Name	Description
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 16-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 16-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18

Table 16-19. DTSTC field descriptions

Bits	Name	Description
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

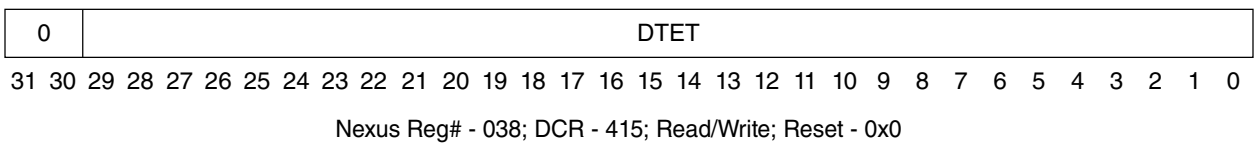


Figure 16-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 16-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control
		00000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10

Table 16-20. DTET field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		X1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

16.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 16-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 16-21. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging

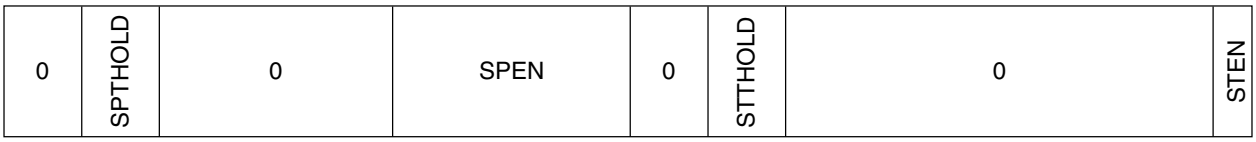
Table continues on the next page...

Table 16-21. WMSK field descriptions (continued)

Bits	Name	Description
		00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

16.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.



Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 16-15. Nexus Overrun Control Register

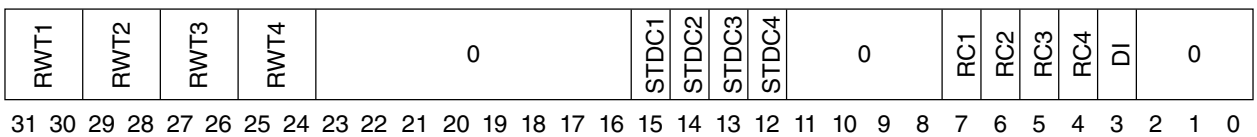
Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 16-16. Nexus Overrun Control Register Fields

16.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.



Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 16-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 16-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1

Table continues on the next page...

Table 16-22. DTC field descriptions (continued)

Bits	Name	Description
		1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z4201n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

16.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 16-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 16-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 16-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 16-21. Data Trace Start Address 4 (DTSA4) register

16.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 16-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 16-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 16-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 16-25. Data Trace End Address 4 (DTEA4) register

Table 16-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 16-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

16.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 16-25.

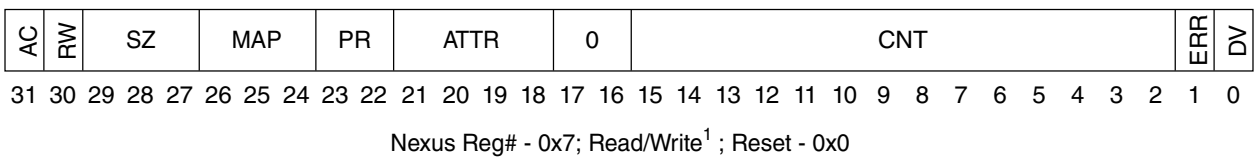


Figure 16-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 16-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority

Table continues on the next page...

Table 16-24. RWCS field description (continued)

Bits	Name	Description
		00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR[2]	Read/Write Access Error (see Table 16-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 16-25)

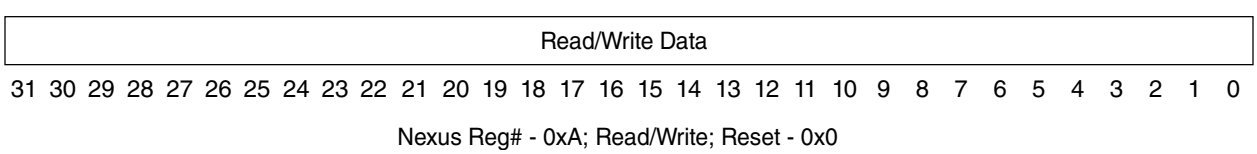
1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 16-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

16.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.



Nexus Reg# - 0xA; Read/Write; Reset - 0x0

Figure 16-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 16-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 16-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 16-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 16-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

16.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

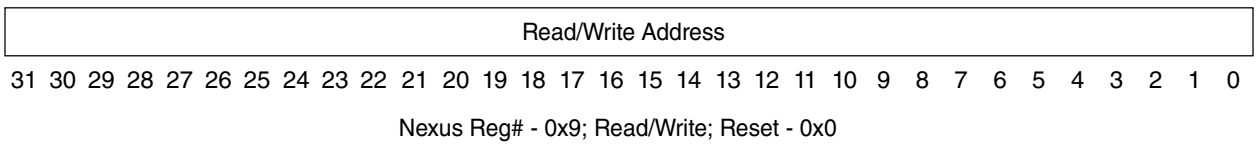


Figure 16-28. Read/Write Access Address (RWA) register

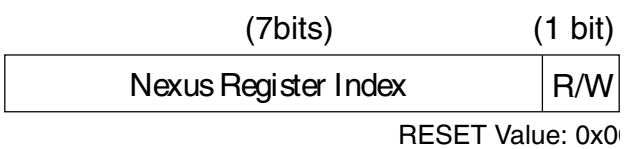
16.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 16-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 16-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:



Nexus Register Index	Selected from values in Table 16-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

16.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 16-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

16.7 Nexus message fields

Nexus messages are composed of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

16.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

16.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

16.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

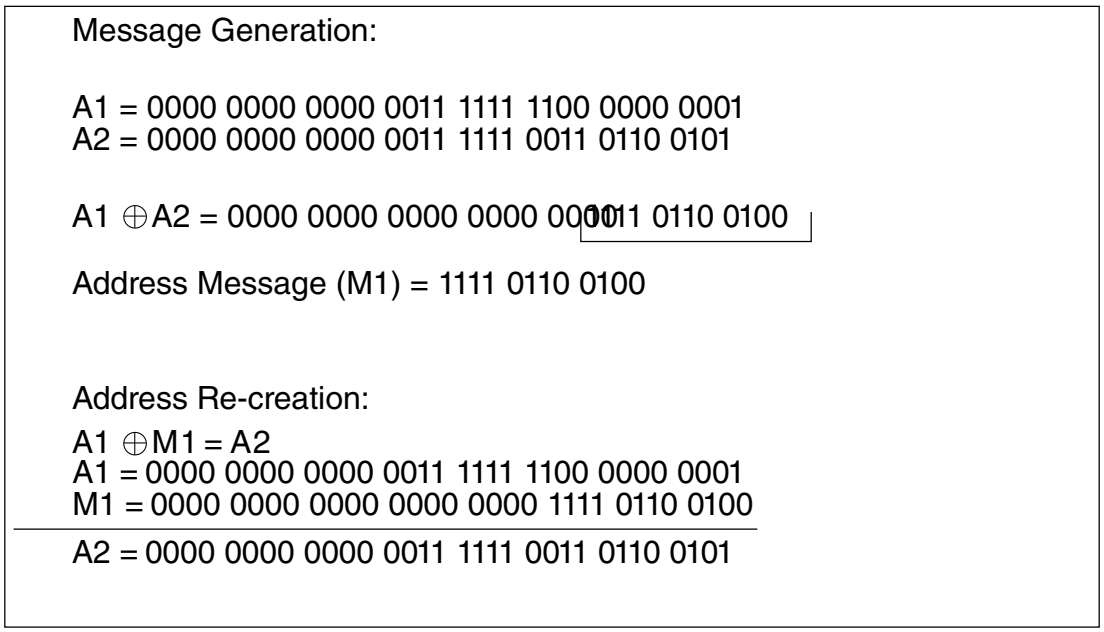


Figure 16-29. Relative address generation and recreation

16.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

16.7.5 Timestamp field (TSTAMP)

Timestamps may be optionally applied to messages. Enabling of timestamping is controlled by the nex_tsen input signal. The timestamping mode (timestamp frequency of occurrence) is controlled by the value of nex_tsmode[0:1] and the timestamp value is based on the nex_timebase[0:29] inputs. These signals are first synchronized to the nex_clk clock domain, and for nex_timebase[0:29], a gray-code to binary value

conversion is performed. The timestamp value is applied to messages as they exit the message queues. Leading zeros in the timestamp value are removed. The table shows the nex_tsmode[0:1] encodings used for the timestamp mode.

Table 16-28. nex_tsmode[0:1] timestamp mode encoding

nex_tsmode[0:1]	Timestamp Mode
00	Timestamp applied to every message
01	Timestamp applied to every 4th message
10	Timestamp applied to every 16th message
11	Timestamp applied to every 64th message

16.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

16.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

16.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

16.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

16.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 16-29 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 16-29.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 16-29. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

16.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

16.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

16.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

16.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

16.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

16.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

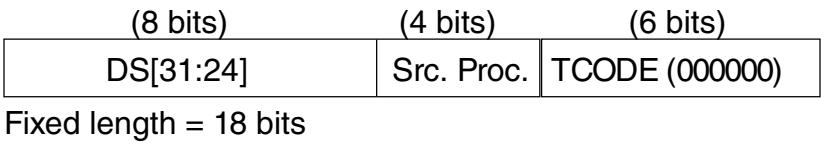


Figure 16-30. Debug Status message format

16.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 16-3 and Table 16-4):

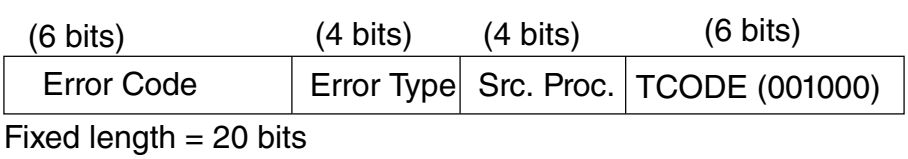


Figure 16-31. Error message format

16.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

16.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

16.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z4201n3) processors contain a *Power ISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

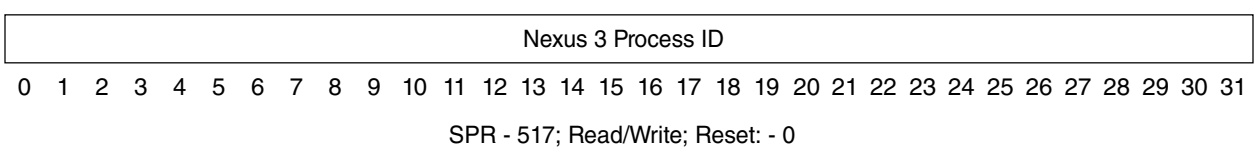


Figure 16-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 16-33. Ownership Trace Message format

16.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z4201n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for

Core (e200z4201n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

16.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

16.12.1.1 Indirect Branch Message instructions

Table 16-30 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 16-30. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

16.12.1.2 Direct Branch Message Instructions

Table 16-31 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 16-31. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

16.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

16.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

16.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

16.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

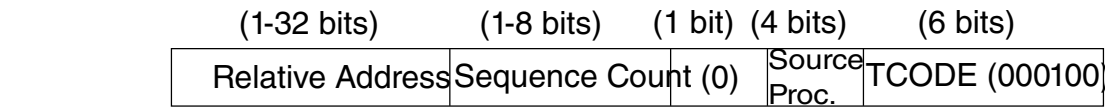
(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 16-34. Indirect branch message (history) format

16.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

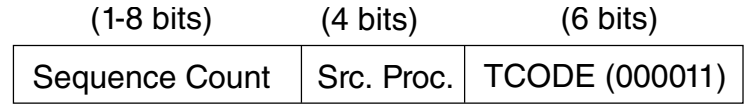


Max length = 51 bits; Min length = 13 bits

Figure 16-35. Indirect branch message format

16.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:



Max length = 18 bits; Min length = 11bits

Figure 16-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

16.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

16.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

16.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 16-32. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

16.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

16.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

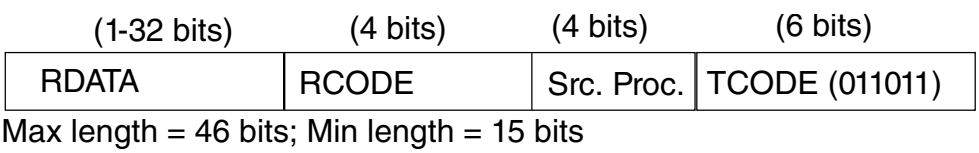


Figure 16-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 16-33. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

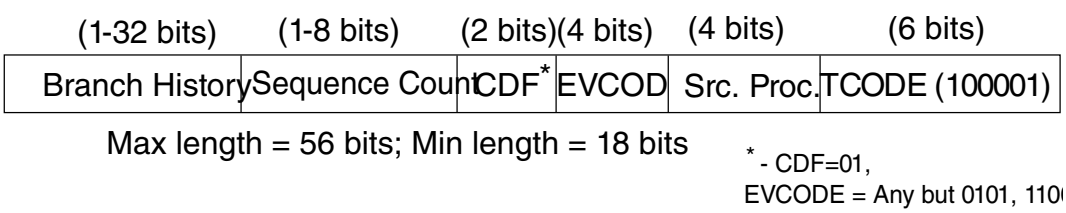
16.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 16-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:



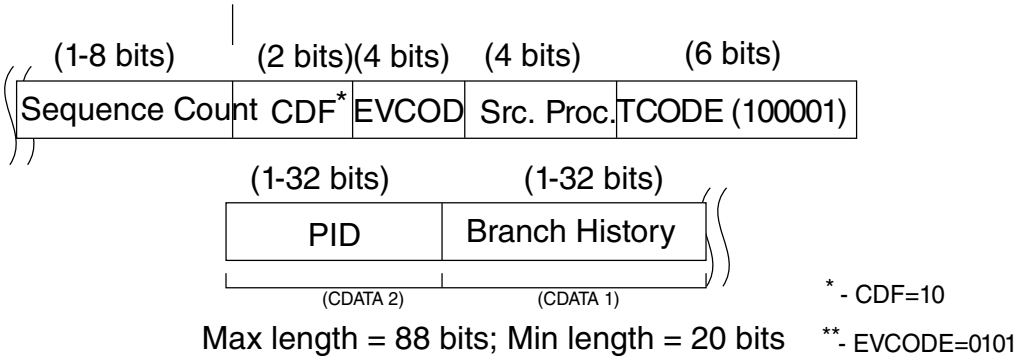


Figure 16-38. Program Correlation message formats

16.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

16.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

16.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 16-34):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message) or suppression, provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in Table 16-34.

Table 16-34. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.

Table continues on the next page...

Table 16-34. Program Trace exception summary (continued)

Exception Condition	Exception Handling
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

16.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 16-34):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

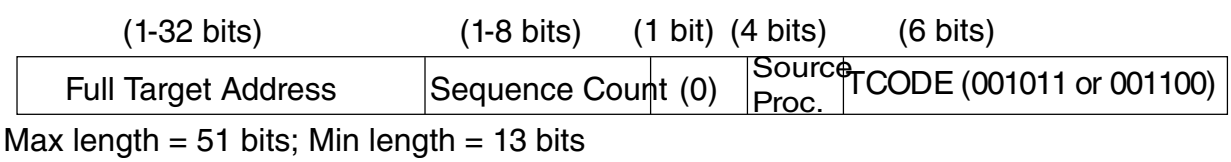


Figure 16-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

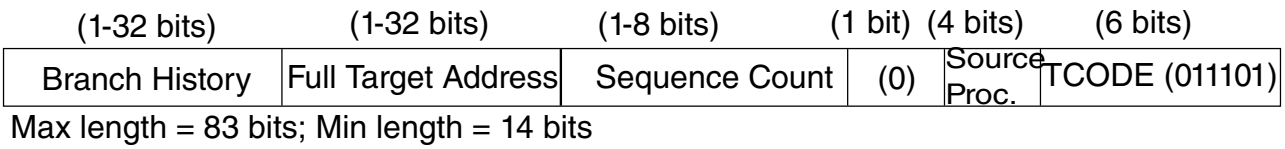


Figure 16-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 16-35.

Table 16-35. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

16.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

16.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

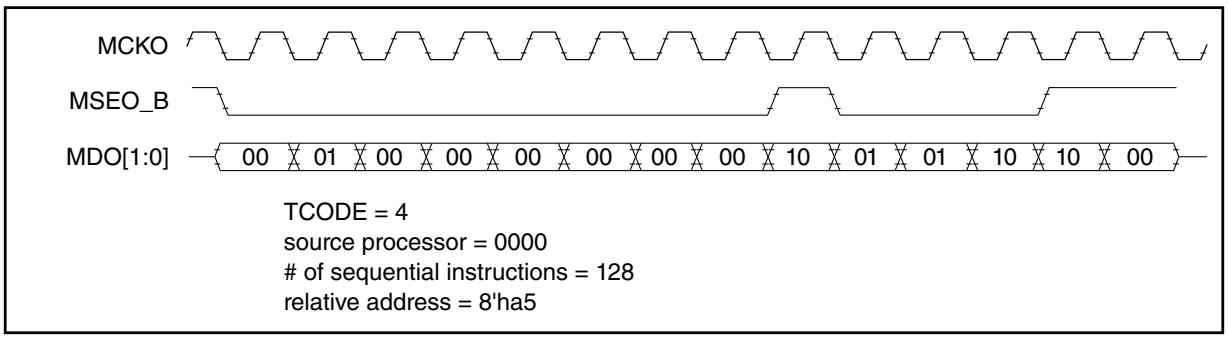


Figure 16-41. Program Trace - Indirect Branch message (Traditional)

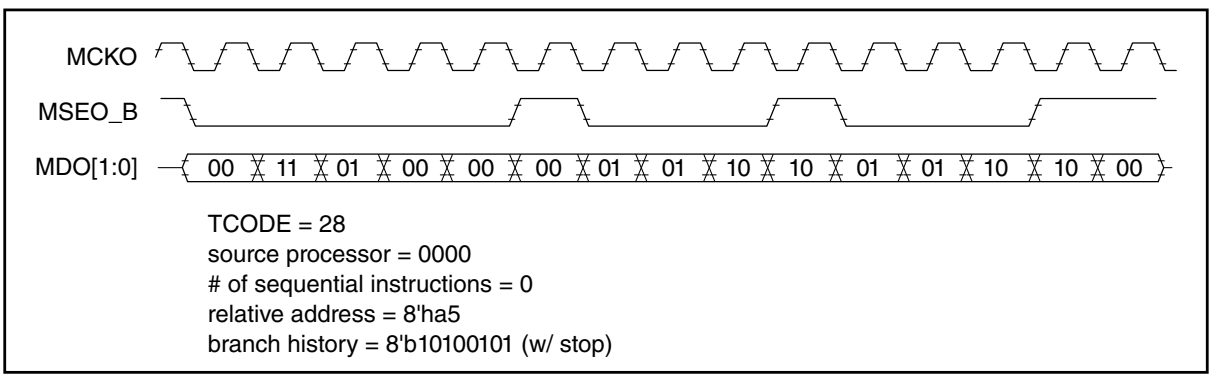


Figure 16-42. Program Trace - Indirect Branch message (history)

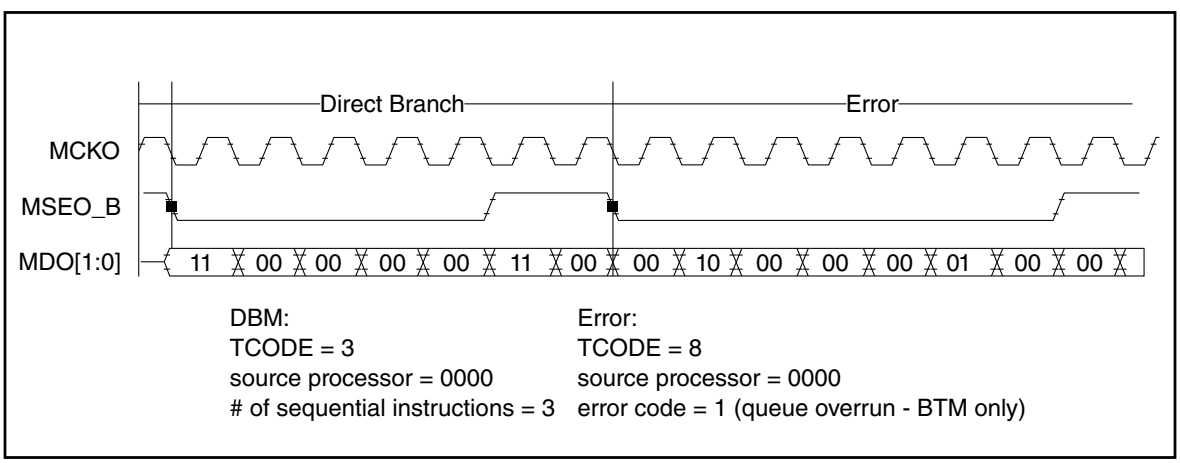


Figure 16-43. Program Trace - Direct Branch (traditional) & Error messages

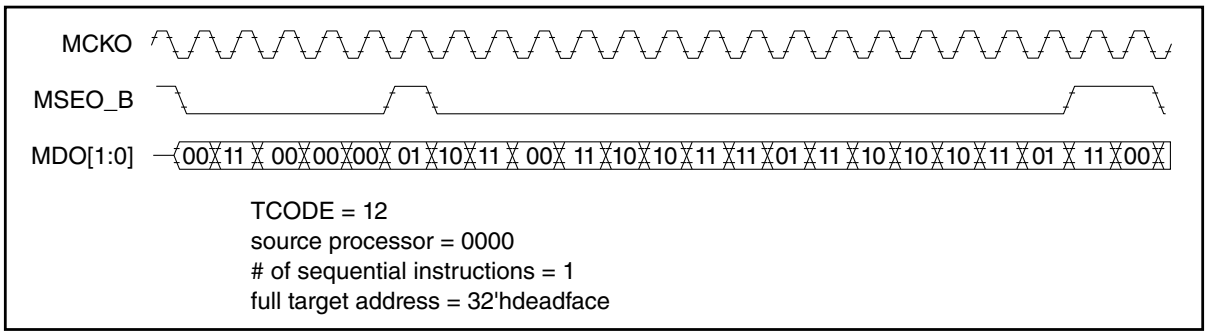


Figure 16-44. Program Trace - Indirect Branch w/ Sync. message

16.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

16.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z4201n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of the following ways:

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

16.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

16.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 16-45. Data Write message format

16.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 16-46. Data Read fessage format

Note

* Core (e200z4201n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z4201n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

16.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 16-36):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message) or suppression, provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

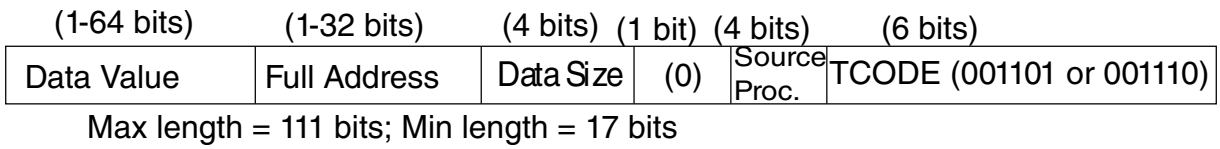


Figure 16-47. Data Write/Read with Sync. message format

Exception conditions that result in Data Trace Synchronization are summarized in Table 16-36.

Table 16-36. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

16.13.3 DTM Operation

16.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

16.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

16.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

16.13.3.4 Bus Cycle special cases

Table 16-37. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$), [1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the

doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z4201n3) Core Debug Support chapter.)

Note

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

16.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

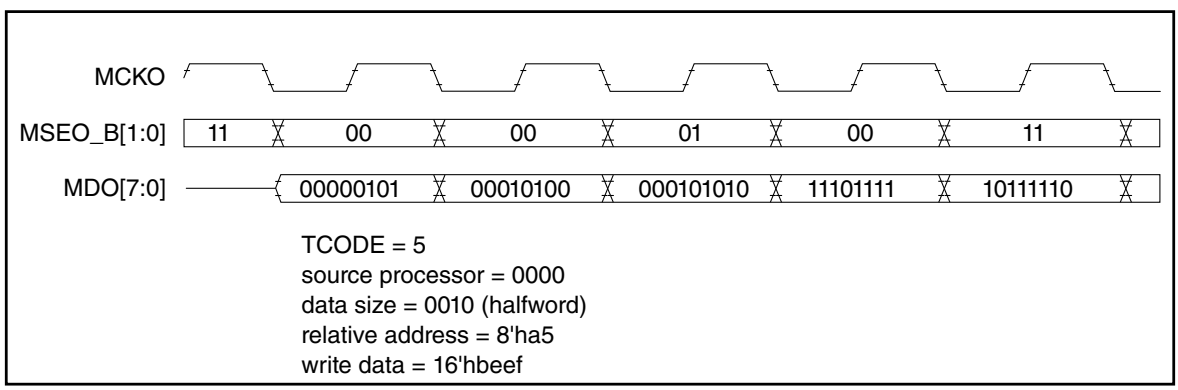


Figure 16-48. Data Trace - Data Write Message

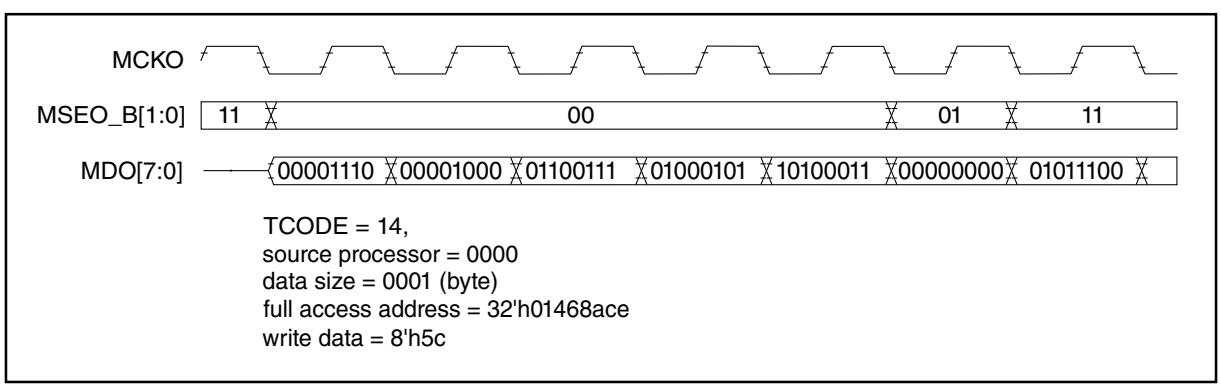


Figure 16-49. Data Trace — Data Read w/ Sync message

16.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

16.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

16.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

16.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z4201n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

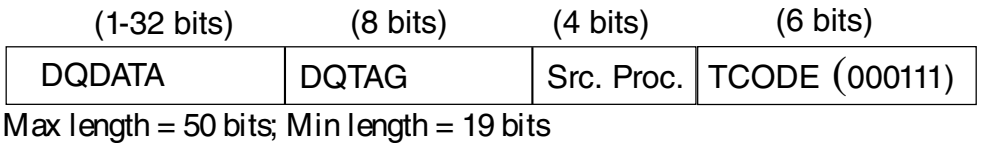


Figure 16-50. Data Acquisition message format

16.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z4201n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 16-40 for details on **evto_b**.

Watchpoint information is messaged out in the following format:

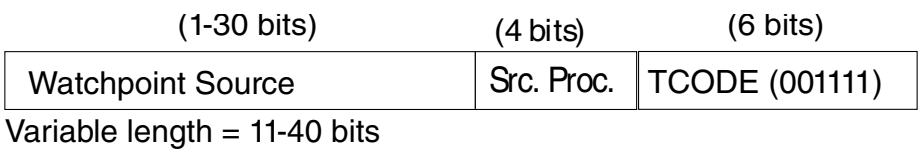


Figure 16-51. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 16-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
000000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	

Table 16-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX	Watchpoint #3 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX	Watchpoint #4 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX	Watchpoint #5 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX	Watchpoint #6 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX	Watchpoint #7 enabled for WTM
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX	Watchpoint #8 enabled for WTM
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX	Watchpoint #9 enabled for WTM
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX	Watchpoint #10 enabled for WTM
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX	Watchpoint #11 enabled for WTM
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX	Watchpoint #12 enabled for WTM
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX	Watchpoint #13 enabled for WTM
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX	Watchpoint #14 enabled for WTM
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX	Watchpoint #15 enabled for WTM
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX	Watchpoint #16 enabled for WTM
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX	Watchpoint #17 enabled for WTM
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX	Watchpoint #18 enabled for WTM
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX	Watchpoint #19 enabled for WTM
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX	Watchpoint #20 enabled for WTM
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #29 enabled for WTM

16.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

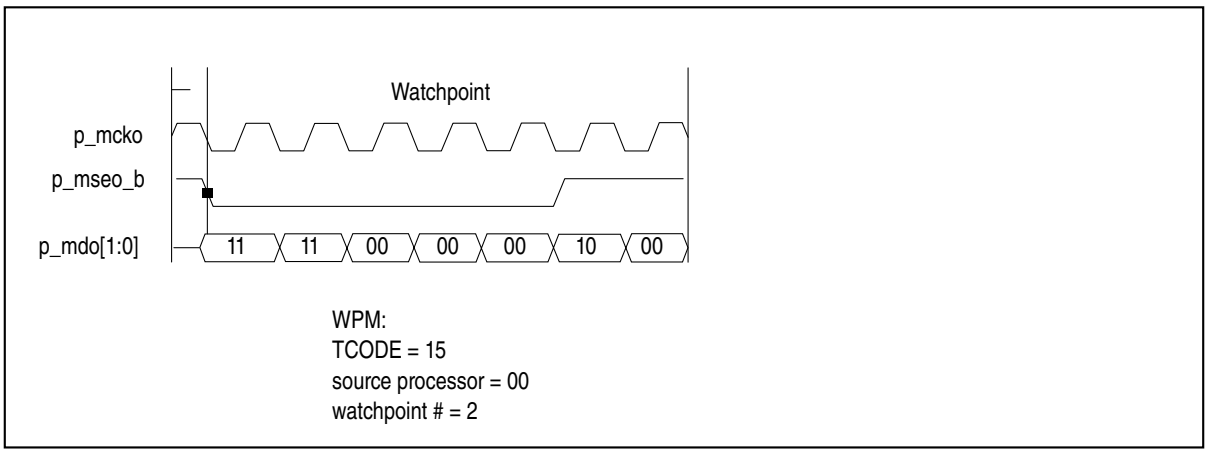


Figure 16-52. Watchpoint Message & Watchpoint Error message

16.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

16.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data (RWD) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 16-40 for details) and clears the ERR bit in the RWCS register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit

will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once the DV bit has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start**,**nex_err_b**, and **nex_rdy_b** pins, as well as the AC, DV, and ERR bits within the RWCS, provide Read/Write Access status to the external development tool.

16.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block then arbitrates for the AHB system bus and transfers the first data value from the RWD register to the memory-mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output is asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS register. Otherwise, the ERR bit is set and the DV bit is cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Clearing the DV bit indicates that the device is ready for the next access in the block transfer, or that an error has occurred.
3. If the AC bit has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS register are not changed when executing a block write access. The original values can be read by the external development tool at any time.

16.16.3 Single read access

1. Initialize the Read/Write Access Address (RWA)register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

An Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 16-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set, and the DV bit will be cleared to indicate an error has occurred. The **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit is cleared in either case.
4. The data can then be read from the Read/Write Access Data register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start**, **nex_err_b**, and **nex_rdy_b** pins as well as the AC, DV, and ERR bits within the RWCS register provide Read/Write Access status to the external development tool.

16.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 16-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS register, and asserts the **nex_rdy_b** pin (see Table 16-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the RWD register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

16.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

16.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

16.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

16.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

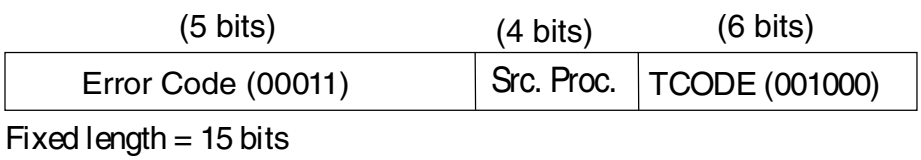


Figure 16-53. Error message format1

16.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

16.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 16-40. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z4201n3) Core Debug Support chapter.

Table 16-39. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 16-40.

Table 16-40. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.

Table continues on the next page...

Table 16-40. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted.
evto_b	O	Event Out (**evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 16-41. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 16-43.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

16.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 16-42. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 16-42. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 16-54 illustrates the state diagram for single pin MSEO transfers.

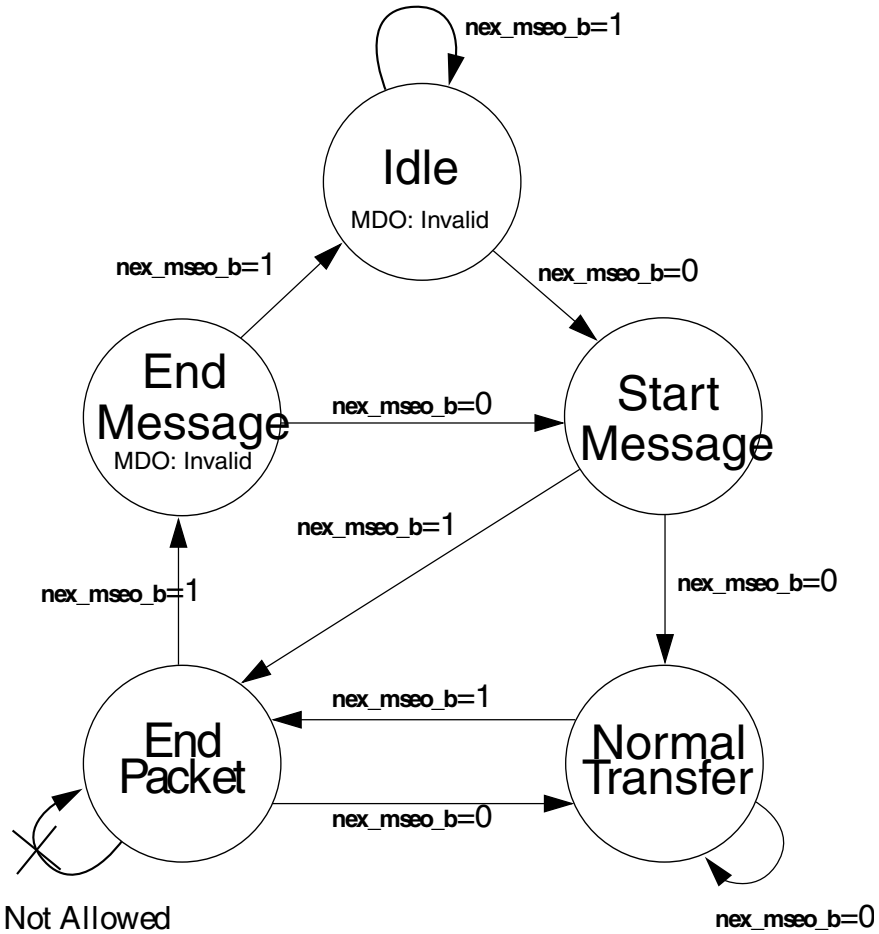


Figure 16-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 16-55 illustrates the state diagram for dual pin MSEO transfers.

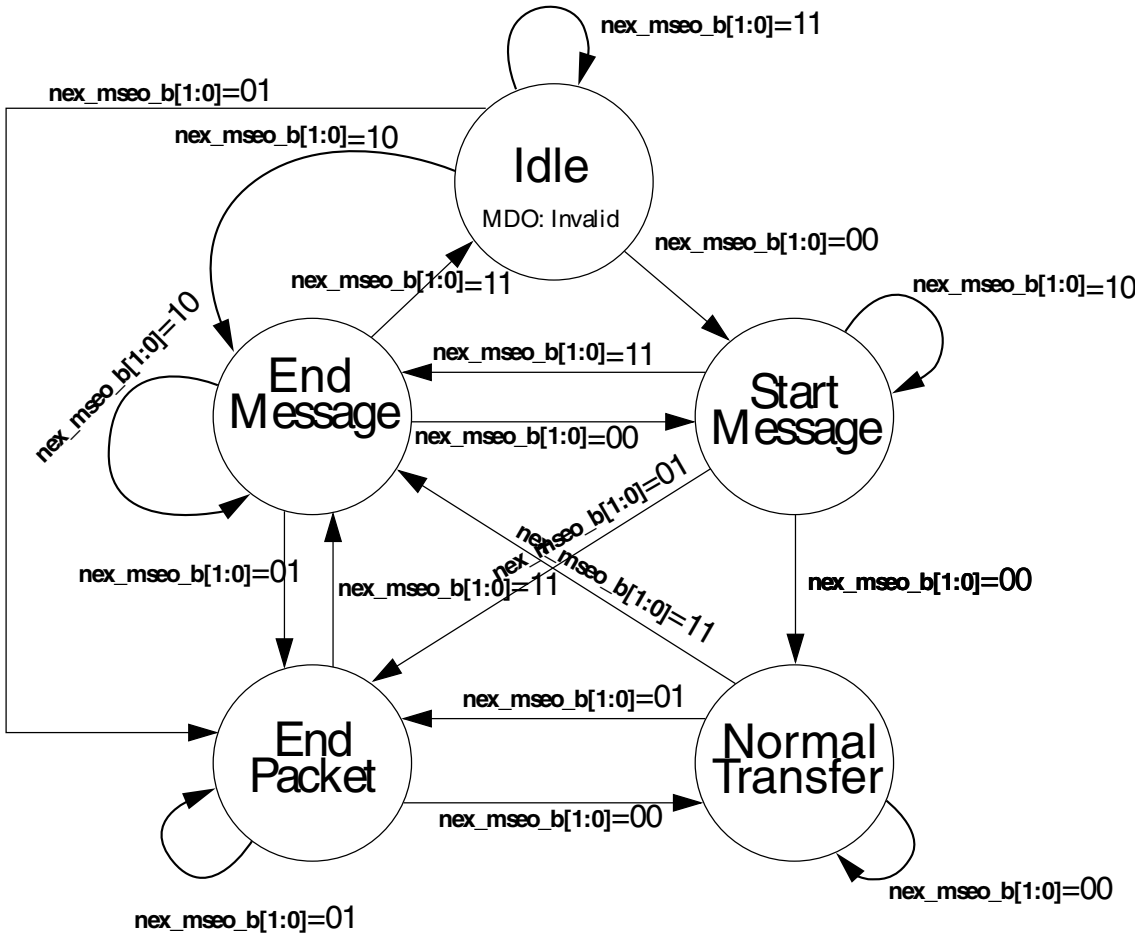


Figure 16-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

16.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

16.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 16-43 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 16-43. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

16.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 16-44 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 16-45 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 16-44. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 16-45. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 16-45. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 16-46 and Table 16-47 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 16-46. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 16-47. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 16-48 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 16-49 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 16-48. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 16-49. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

16.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

16.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

16.22.1 JTAG sequence for accessing internal Nexus registers

Table 16-50. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

16.22.2 JTAG sequence for read access of memory-mapped resources

Table 16-51. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

16.22.3 JTAG sequence for write access of memory-mapped resources

Table 16-52. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b**pin
8	—	If CNT > 0, go back to Step #5

Chapter 17
z7 Nexus Module

17.1 Introduction

The e200z7260n3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

17.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

17.1.2 Terms and Definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 17-1. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

17.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

17.1.4 Functional Block Diagram

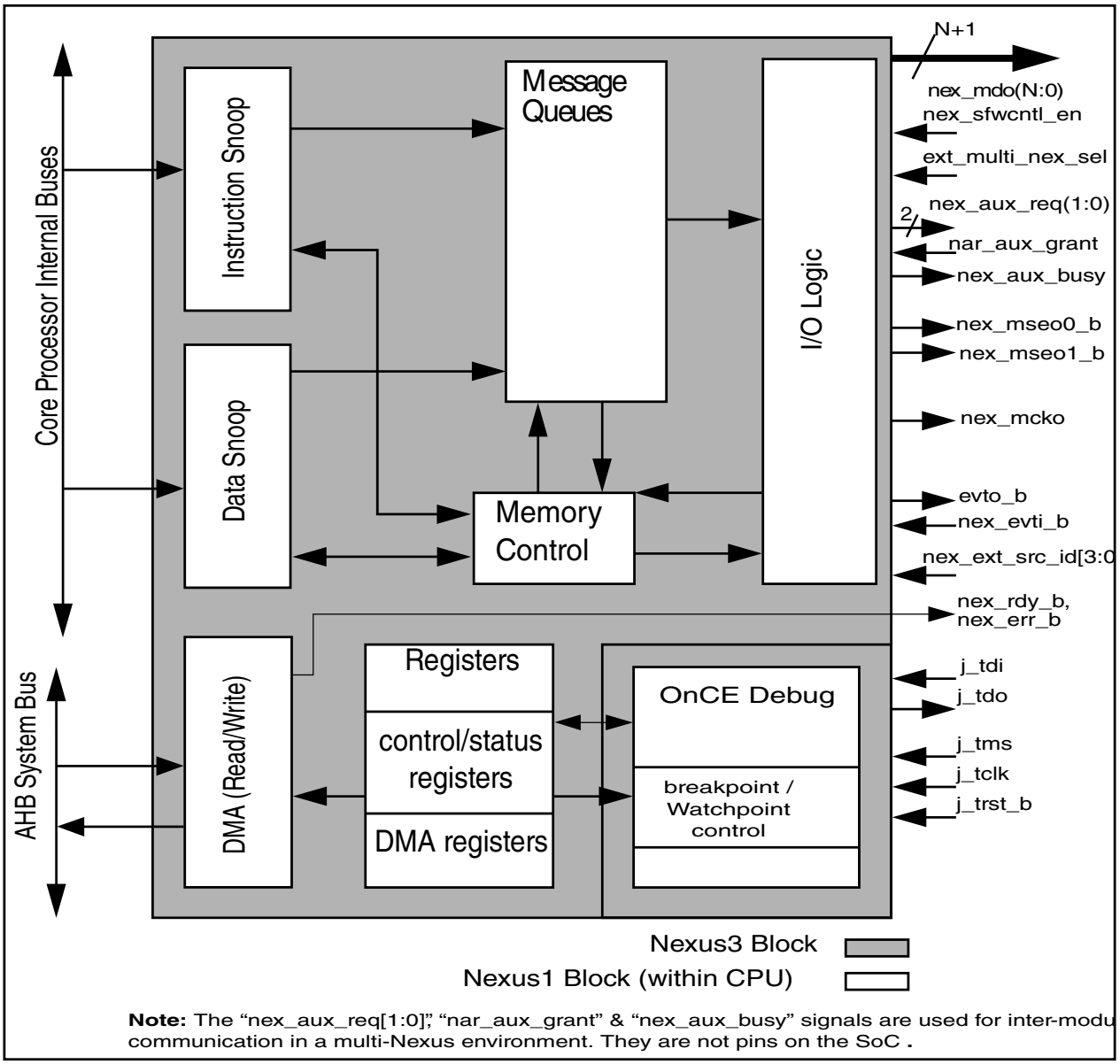


Figure 17-1. Nexus 3 Functional Block Diagram

17.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

17.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

17.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 17-2.

Table 17-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 17-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 17-7)

Table continues on the next page...

Table 17-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 17-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace -	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier

Table continues on the next page...

Table 17-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Read Message w/ Sync	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 17-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 17-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 17-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)

Table continues on the next page...

Table 17-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 17-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 17-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 17-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 17-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost

Table continues on the next page...

Table 17-4. Error Type (ETYPE) Encoding (TCODE = 8) (continued)

Error Type	Description
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 17-5 shows the encodings used for resource codes for certain messages.

Table 17-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 17-6 shows the event code encodings used for certain messages.

Table 17-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 17-7 shows the data trace size encodings used for certain messages.

Table 17-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)

Table continues on the next page...

Table 17-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

17.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 17-8. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	

Table continues on the next page...

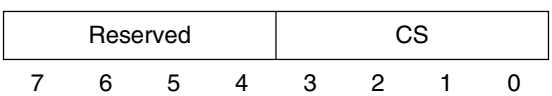
Table 17-8. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

17.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# - 0x1; Read-only; Reset - 0x0

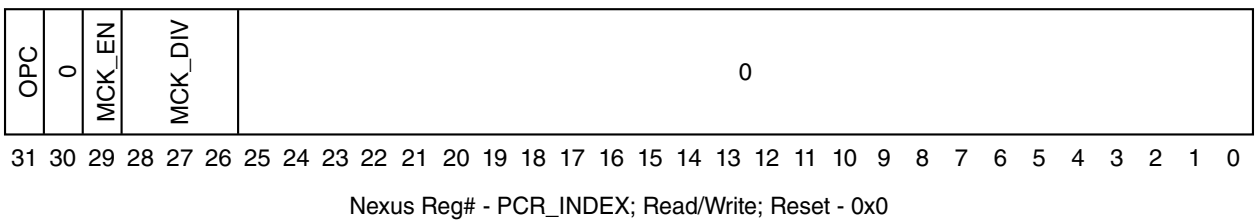
Figure 17-2. Client Select Control (CSC) register

Table 17-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

17.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 17-3. Port Configuration Register

Table 17-10. PCR field descriptions

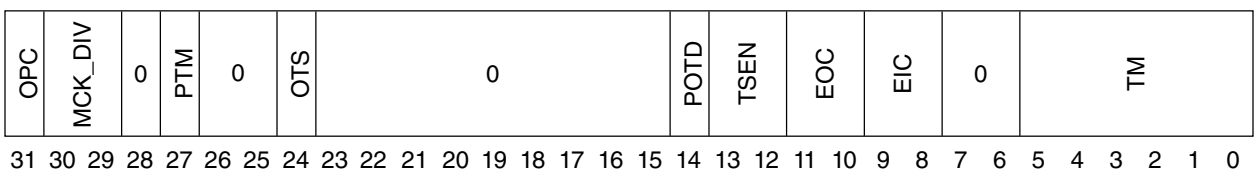
Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

17.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 17-4 and its fields are described in Table 17-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 17-4. Development Control Register 1

Table 17-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality

Table continues on the next page...

Table 17-11. DC1 field descriptions (continued)

Bits	Name	Description
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.

Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

17.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z7260n3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 17-5 and its fields are described in Table 17-12.

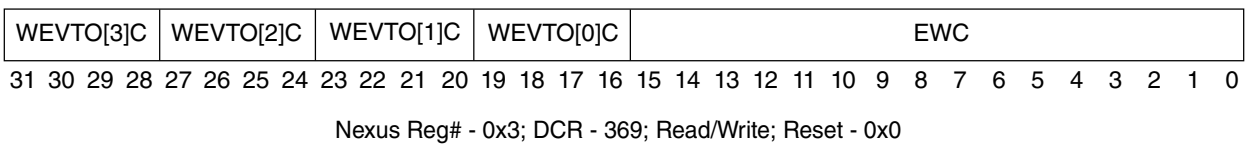


Figure 17-5. Development Control Register 2

Table 17-12. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[3]** 0001 Watchpoint #0 triggers **nex_wevto[3]** 0010 Watchpoint #1 triggers **nex_wevto[3]** 0011 Watchpoint #2 triggers **nex_wevto[3]** 0100 Watchpoint #3 triggers **nex_wevto[3]** 0101 Watchpoint #4 triggers **nex_wevto[3]** 0110 Watchpoint #5 triggers **nex_wevto[3]** 0111 Watchpoint #6 triggers **nex_wevto[3]** 1000 Watchpoint #7 triggers **nex_wevto[3]** 1001 Watchpoint #8 triggers **nex_wevto[3]** 1010 Watchpoint #9 triggers **nex_wevto[3]** 1011 Watchpoint #10 triggers **nex_wevto[3]** 1100 Watchpoint #11 triggers **nex_wevto[3]** 1101 Watchpoint #12 triggers **nex_wevto[3]** 1110 Watchpoint #13 triggers **nex_wevto[3]** 1111 Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[2]** 0001 Watchpoint #0 triggers **nex_wevto[2]** 0010 Watchpoint #1 triggers **nex_wevto[2]** 0011 Watchpoint #2 triggers **nex_wevto[2]** 0100 Watchpoint #3 triggers **nex_wevto[2]**

Table continues on the next page...

Table 17-12. DC2 field descriptions (continued)

Bits	Name	Description
		0101 Watchpoint #4 triggers **nex_wevto[2]**
		0110 Watchpoint #5 triggers **nex_wevto[2]**
		0111 Watchpoint #6 triggers **nex_wevto[2]**
		1000 Watchpoint #7 triggers **nex_wevto[2]**
		1001 Watchpoint #8 triggers **nex_wevto[2]**
		1010 Watchpoint #9 triggers **nex_wevto[2]**
		1011 Watchpoint #10 triggers **nex_wevto[2]**
		1100 Watchpoint #11 triggers **nex_wevto[2]**
		1101 Watchpoint #12 triggers **nex_wevto[2]**
		1110 Watchpoint #13 triggers **nex_wevto[2]**
		1111 Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[1]**
		0001 Watchpoint #0 triggers **nex_wevto[1]**
		0010 Watchpoint #1 triggers **nex_wevto[1]**
		0011 Watchpoint #2 triggers **nex_wevto[1]**
		0100 Watchpoint #3 triggers **nex_wevto[1]**
		0101 Watchpoint #4 triggers **nex_wevto[1]**
		0110 Watchpoint #5 triggers **nex_wevto[1]**
		0111 Watchpoint #6 triggers **nex_wevto[1]**
		1000 Watchpoint #7 triggers **nex_wevto[1]**
		1001 Watchpoint #8 triggers **nex_wevto[1]**
		1010 Watchpoint #9 triggers **nex_wevto[1]**
		1011 Watchpoint #10 triggers **nex_wevto[1]**
		1100 Watchpoint #11 triggers **nex_wevto[1]**
		1101 Watchpoint #12 triggers **nex_wevto[1]**
		1110 Watchpoint #13 triggers **nex_wevto[1]**
		1111 Watchpoint #14 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #0-14 trigger **nex_wevto[0]**
		0001 Watchpoint #0 triggers **nex_wevto[0]**
		0010 Watchpoint #1 triggers **nex_wevto[0]**
		0011 Watchpoint #2 triggers **nex_wevto[0]**
		0100 Watchpoint #3 triggers **nex_wevto[0]**
		0101 Watchpoint #4 triggers **nex_wevto[0]**
		0110 Watchpoint #5 triggers **nex_wevto[0]**
		0111 Watchpoint #6 triggers **nex_wevto[0]**

Table continues on the next page...

Table 17-12. DC2 field descriptions (continued)

Bits	Name	Description
		1000 Watchpoint #7 triggers **nex_wevto[0]**
		1001 Watchpoint #8 triggers **nex_wevto[0]**
		1010 Watchpoint #9 triggers **nex_wevto[0]**
		1011 Watchpoint #10 triggers **nex_wevto[0]**
		1100 Watchpoint #11 triggers **nex_wevto[0]**
		1101 Watchpoint #12 triggers **nex_wevto[0]**
		1110 Watchpoint #13 triggers **nex_wevto[0]**
		1111 Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration[1]
		0000000000000000 No Watchpoints #0-15 trigger **evto_b**
		XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **evto_b**
		XXXXXXXXXXXXXX1X Watchpoint #1 triggers **evto_b**
		XXXXXXXXXXXXX1XX Watchpoint #2 triggers **evto_b**
		XXXXXXXXXXXX1XXX Watchpoint #3 triggers **evto_b**
		XXXXXXXXXXX1XXXX Watchpoint #4 triggers **evto_b**
		XXXXXXXXXX1XXXXX Watchpoint #5 triggers **evto_b**
		XXXXXXXXX1XXXXXX Watchpoint #6 triggers **evto_b**
		XXXXXXXX1XXXXXXX Watchpoint #7 triggers **evto_b**
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **evto_b**
		XXXXXX1XXXXXXXXX Watchpoint #9 triggers **evto_b**
		XXXXX1XXXXXXXXXX Watchpoint #10 triggers **evto_b**
		XXXX1XXXXXXXXXXX Watchpoint #11 triggers **evto_b**
		XXX1XXXXXXXXXXXX Watchpoint #12 triggers **evto_b**
		XX1XXXXXXXXXXXXX Watchpoint #13 triggers **evto_b**
		X1XXXXXXXXXXXXXX Watchpoint #14 triggers **evto_b**
		1XXXXXXXXXXXXXXX Watchpoint #15 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 17-6 and its fields are described in Table 17-13.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 17-6. Development Control 3 (DCR3) register

Table 17-13. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15-#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 1101 - 1111 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15-#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 1101 - 1111 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15-#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]**

Table continues on the next page...

Table 17-13. DC3 field descriptions (continued)

Bits	Name	Description
		0110 Watchpoint #20 triggers **nex_wevto[1]**
		0111 Watchpoint #21 triggers **nex_wevto[1]**
		1000 Watchpoint #22 triggers **nex_wevto[1]**
		1001 Watchpoint #23 triggers **nex_wevto[1]**
		1010 Watchpoint #24 triggers **nex_wevto[1]**
		1011 Watchpoint #25 triggers **nex_wevto[1]**
		1100 Watchpoint #26 triggers **nex_wevto[1]**
		1101 Watchpoint #27 triggers **nex_wevto[1]**
		1110 Watchpoint #28 triggers **nex_wevto[1]**
		1111 Watchpoint #29 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration
		0000 No Watchpoints #15-#29 trigger **nex_wevto[0]**
		0001 Watchpoint #15 triggers **nex_wevto[0]**
		0010 Watchpoint #16 triggers **nex_wevto[0]**
		0011 Watchpoint #17 triggers **nex_wevto[0]**
		0100 Watchpoint #18 triggers **nex_wevto[0]**
		0101 Watchpoint #19 triggers **nex_wevto[0]**
		0110 Watchpoint #20 triggers **nex_wevto[0]**
		0111 Watchpoint #21 triggers **nex_wevto[0]**
		1000 Watchpoint #22 triggers **nex_wevto[0]**
		1001 Watchpoint #23 triggers **nex_wevto[0]**
		1010 Watchpoint #24 triggers **nex_wevto[0]**
		1011 Watchpoint #25 triggers **nex_wevto[0]**
		1100 Watchpoint #26 triggers **nex_wevto[0]**
		1101 Watchpoint #27 triggers **nex_wevto[0]**
		1110 Watchpoint #28 triggers **nex_wevto[0]**
		1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1]
		00000000000000 No Watchpoints #16-#29 trigger **evto_b**
		XXXXXXXXXXXXX1 Watchpoint #16 triggers **evto_b**
		XXXXXXXXXXXX1X Watchpoint #17 triggers **evto_b**
		XXXXXXXXXXX1XX Watchpoint #18 triggers **evto_b**
		XXXXXXXXXX1XXX Watchpoint #19 triggers **evto_b**
		XXXXXXXXX1XXXX Watchpoint #20 triggers **evto_b**
		XXXXXXXX1XXXXX Watchpoint #21 triggers **evto_b**
		XXXXXXX1XXXXXX Watchpoint #22 triggers **evto_b**

Table 17-13. DC3 field descriptions

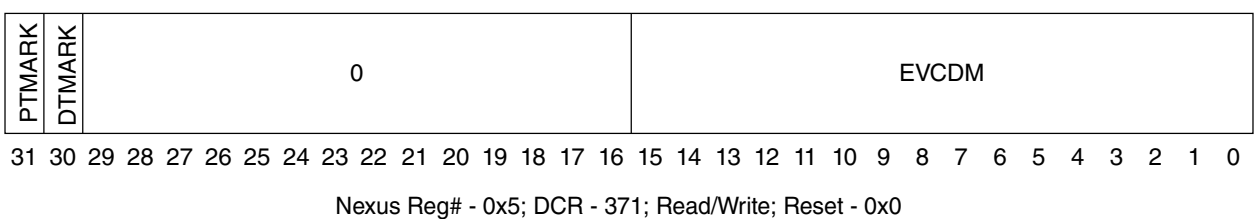
Bits	Name	Description
		XXXXXX1XXXXXXX Watchpoint #23 triggers **evto_b**
		XXXXX1XXXXXXXX Watchpoint #24 triggers **evto_b**
		XXXX1XXXXXXXXX Watchpoint #25 triggers **evto_b**
		XXX1XXXXXXXXXX Watchpoint #26 triggers **evto_b**
		XX1XXXXXXXXXXX Watchpoint #27 triggers **evto_b**
		X1XXXXXXXXXXXX Watchpoint #28 triggers **evto_b**
		1XXXXXXXXXXXXX Watchpoint #29 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

17.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 17-7 and its fields are described in Table 17-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

Figure 17-7. Development Control 4 (DC4) register

Table 17-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask[1] 0000000000000000 No EVCODEs masked for Program Correlation Messages

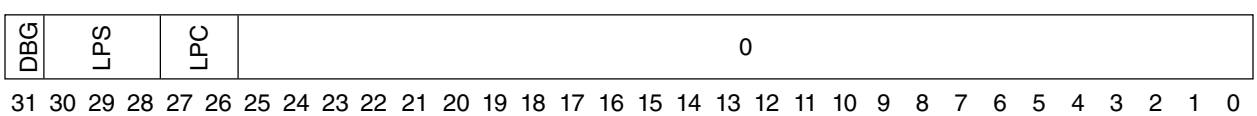
Table 17-14. DC4 field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages
		XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages
		XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages
		XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages
		XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages
		XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages
		XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages
		XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages
		XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages
		XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages
		XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages
		XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages
		XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages
		XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages
		X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages
		1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

1. Refer to Table 5 for implemented EVCODEs.
1. Refer to Table 5 for implemented EVCODEs.

17.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 17-8. Development Status Register

Table 17-15. DS field description

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)

Table continues on the next page...

Table 17-15. DS field description (continued)

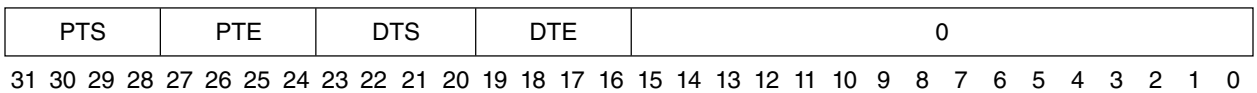
Bits	Name	Description
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

17.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers

for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z7260n3) Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 17-9. Watchpoint Trigger (WT) Register

Table 17-16 details the Watchpoint Trigger register fields.

Table 17-16. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13

Table continues on the next page...

Table 17-16. WT field descriptions (continued)

Bits	Name	Description
		1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

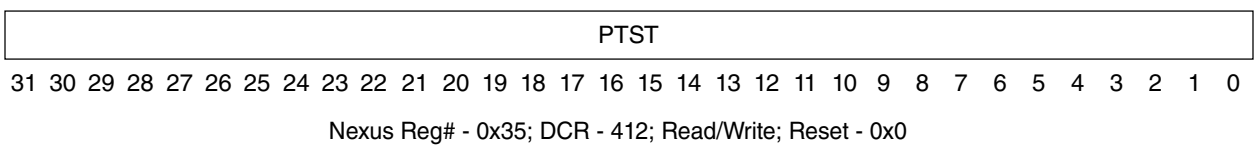


Figure 17-10. Program Trace Start Trigger Control (PTSTC) register

Table 17-17 details the PTSTC register fields.

Table 17-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13

Table 17-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		X1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 17-11. Program Trace End Trigger Control (PTETC) register

Table 17-18 details the PTETC register fields.

Table 17-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4

Table 17-18. PTETC field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 17-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 17-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 038; DCR - 415; Read/Write; Reset - 0x0

Figure 17-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 17-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27

Table 17-20. DTET field descriptions

Bits	Name	Description
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

17.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 17-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 17-21. WMSK field descriptions

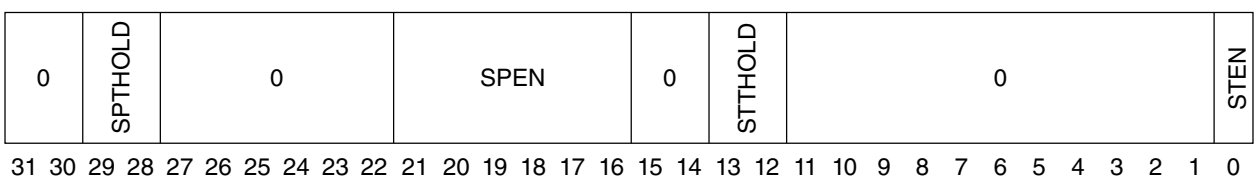
Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM

Table 17-21. WMSK field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

17.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.



Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 17-15. Nexus Overrun Control Register

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 17-16. Nexus Overrun Control Register Fields

17.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1	RWT2	RWT3	RWT4	0	STDC1	STDC2	STDC3	STDC4	0	RC1	RC2	RC3	RC4	DI	0
31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16	15	14	13	12	11 10 9 8	7	6	5	4	3	2 1 0

Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 17-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 17-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range

Table continues on the next page...

Table 17-22. DTC field descriptions (continued)

Bits	Name	Description
		1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z7260n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

17.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 17-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 17-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 17-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 17-21. Data Trace Start Address 4 (DTSA4) register

17.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 17-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 17-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 17-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 17-25. Data Trace End Address 4 (DTEA4) register

Table 17-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 17-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

17.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 17-25.

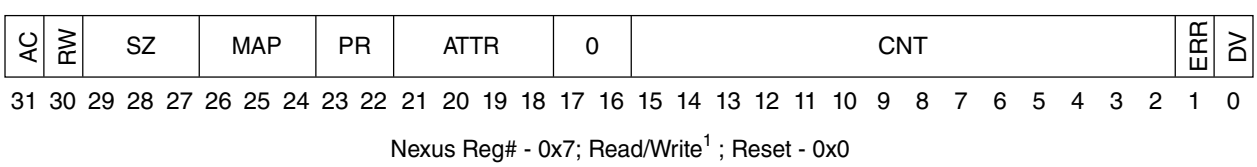


Figure 17-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 17-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR[2]	Read/Write Access Error (see Table 17-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 17-25)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 17-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

17.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

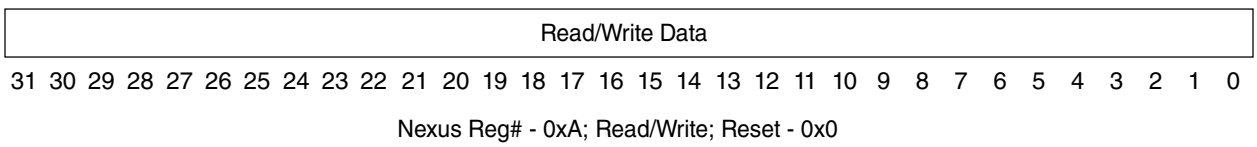


Figure 17-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 17-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 17-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 17-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 17-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

17.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

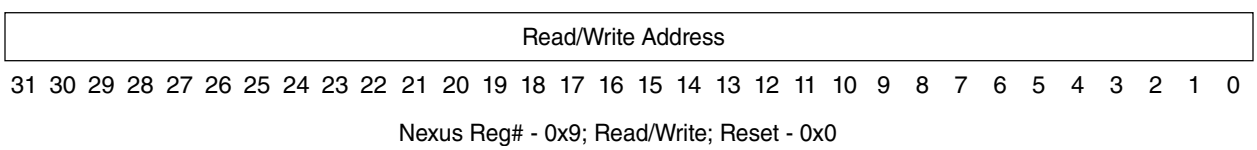


Figure 17-28. Read/Write Access Address (RWA) register

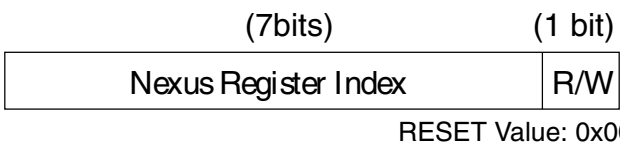
17.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 17-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 17-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:



Nexus Register Index	Selected from values in Table 17-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

17.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no

possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 17-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

17.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

17.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

17.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

17.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

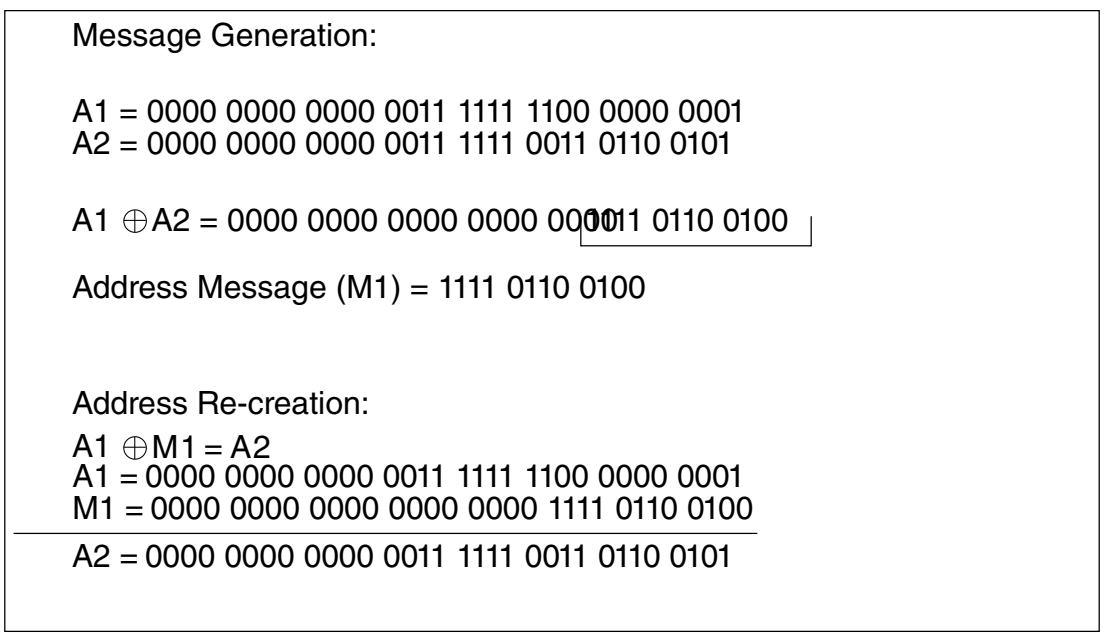


Figure 17-29. Relative address generation and recreation

17.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

17.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

17.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

17.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

17.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

17.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 17-28 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 17-28.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 17-28. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM

Table continues on the next page...

Table 17-28. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

17.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

17.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

17.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

17.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

17.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

17.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

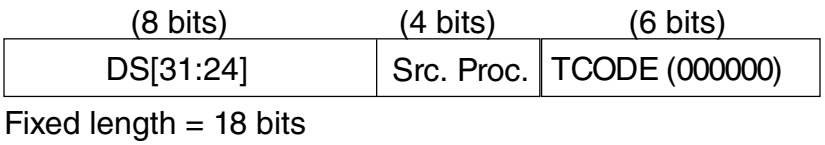


Figure 17-30. Debug Status message format

17.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 17-3 and Table 17-4):

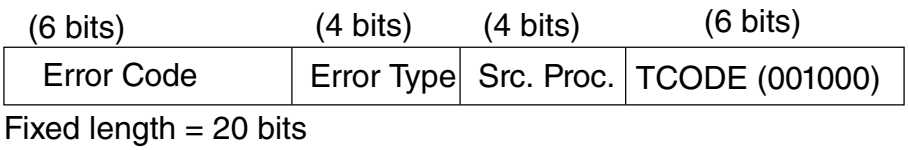


Figure 17-31. Error message format

17.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

17.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

17.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z7260n3) processors contain a Power ISA 2.06 defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

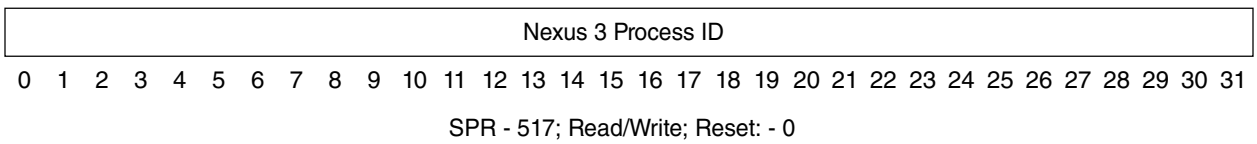


Figure 17-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 17-33. Ownership Trace Message format

17.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z7260n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z7260n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

17.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a

history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

17.12.1.1 Indirect Branch Message instructions

Table 17-29 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 17-29. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

17.12.1.2 Direct Branch Message Instructions

Table 17-30 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 17-30. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

17.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1)

indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

17.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

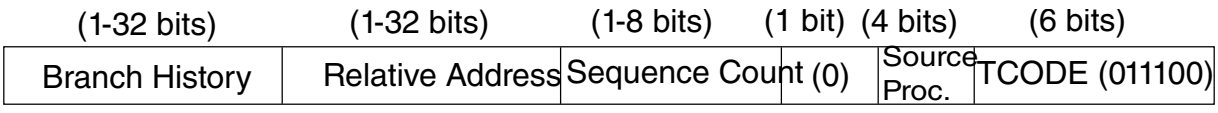
Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

17.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

17.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

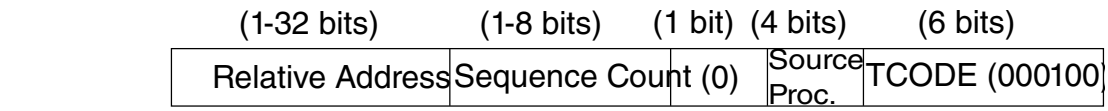


Max length = 83 bits; Min length = 14 bits

Figure 17-34. Indirect branch message (history) format

17.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

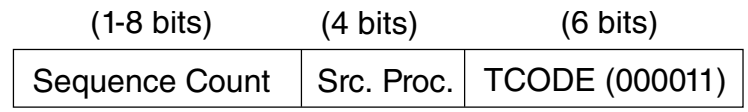


Max length = 51 bits; Min length = 13 bits

Figure 17-35. Indirect branch message format

17.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:



Max length = 18 bits; Min length = 11bits

Figure 17-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

17.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

17.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

17.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 17-31. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**

Table continues on the next page...

Table 17-31. Branch/Predicate history events (continued)

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

17.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

17.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

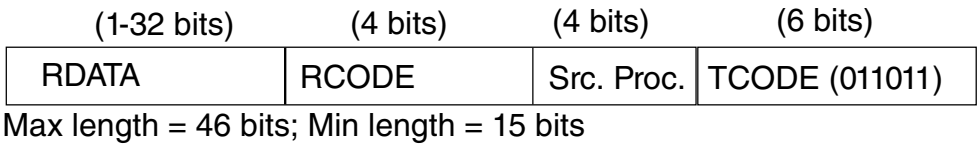


Figure 17-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 17-32. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

17.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 17-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

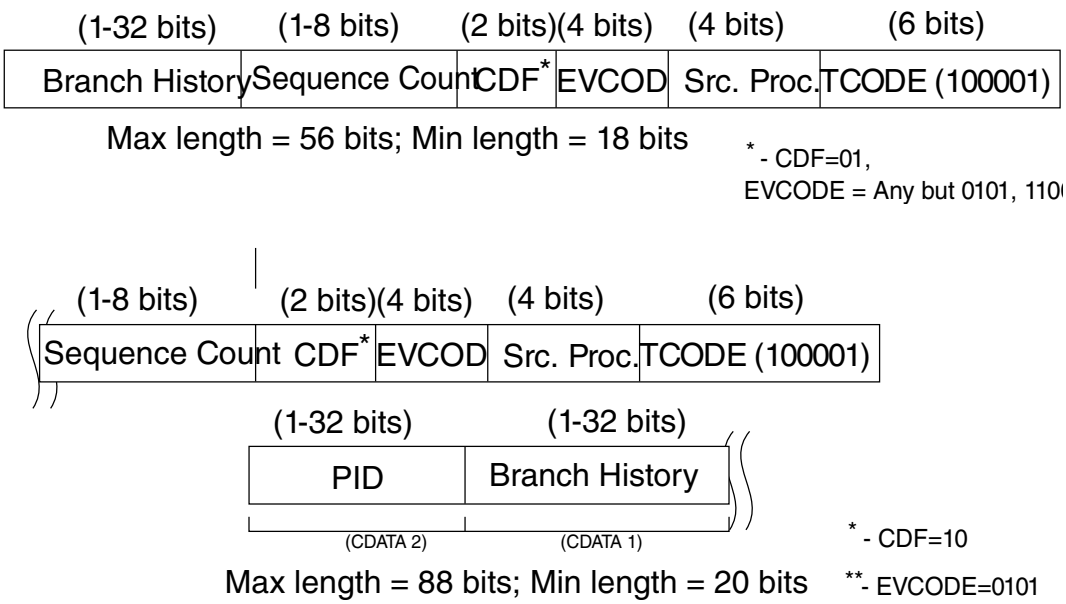


Figure 17-38. Program Correlation message formats

17.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

17.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

17.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 17-33):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in Table 17-33.

Table 17-33. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

17.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 17-33):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

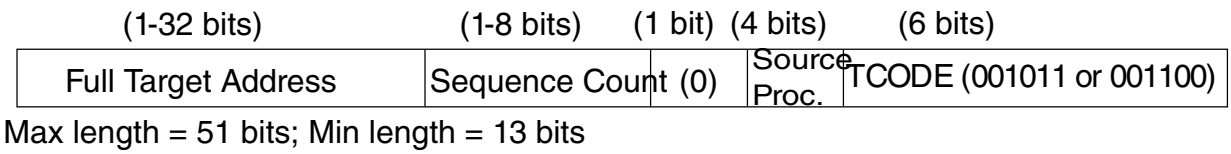


Figure 17-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

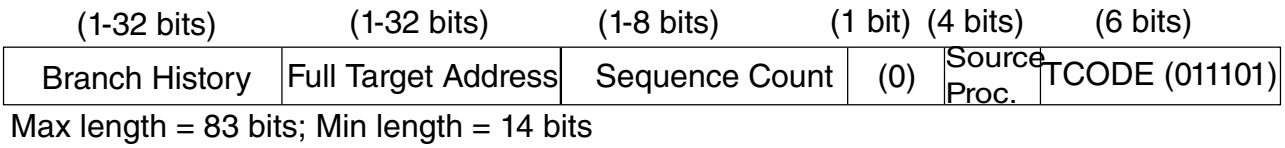


Figure 17-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 17-34.

Table 17-34. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

17.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

17.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

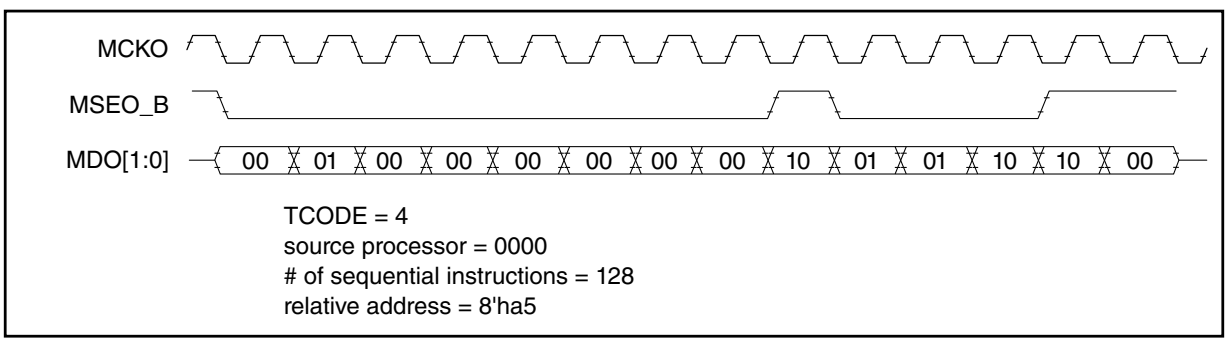


Figure 17-41. Program Trace - Indirect Branch message (Traditional)

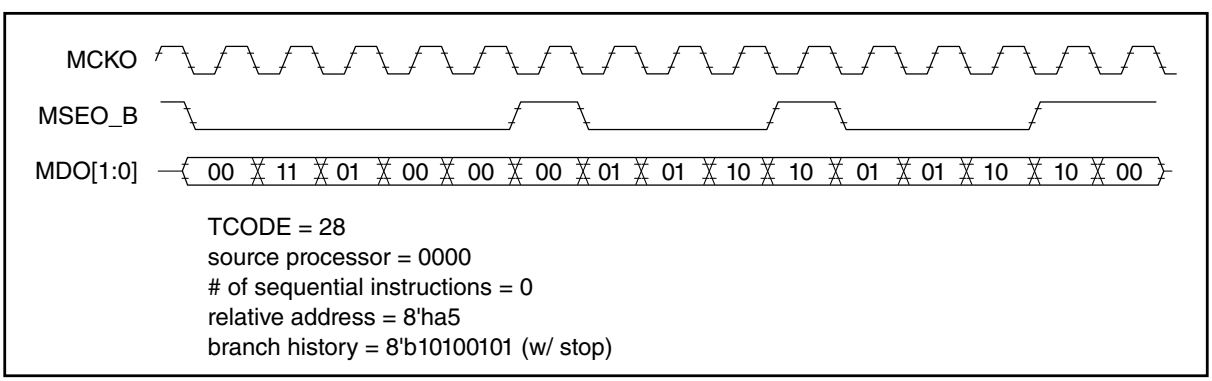


Figure 17-42. Program Trace - Indirect Branch message (history)

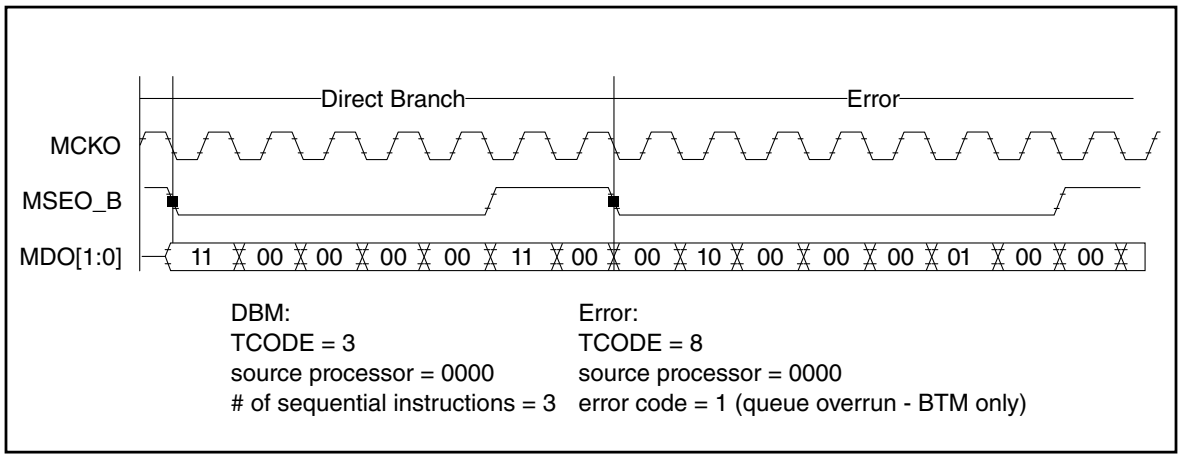


Figure 17-43. Program Trace - Direct Branch (traditional) & Error messages

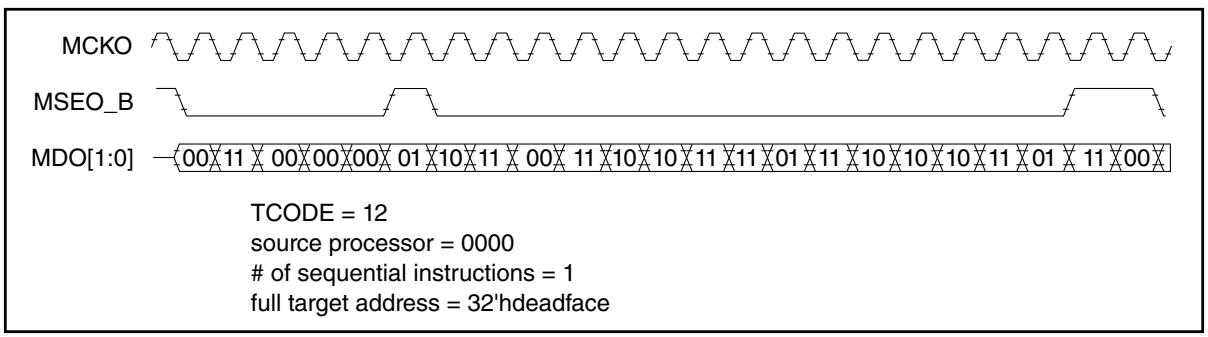


Figure 17-44. Program Trace - Indirect Branch w/ Sync. message

17.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

17.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z7260n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

17.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

17.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 17-45. Data Write message format

17.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 17-46. Data Read fessage format

Note

* Core (e200z7260n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z7260n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

17.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 17-35):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.

- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

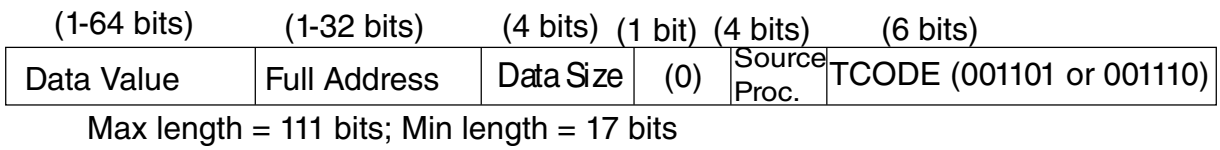


Figure 17-47. Data Write/Read with Sync. message format

Exception conditions that result in Data Trace Synchronization are summarized in Table 17-35.

Table 17-35. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

17.13.3 DTM Operation

17.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

17.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

17.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

17.13.3.4 Bus Cycle special cases

Table 17-36. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$), [1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)

Table continues on the next page...

Table 17-36. Bus Cycle Special Cases (continued)

Special Case	Action
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z7260n3) Core Debug Support chapter.)

Note

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

17.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

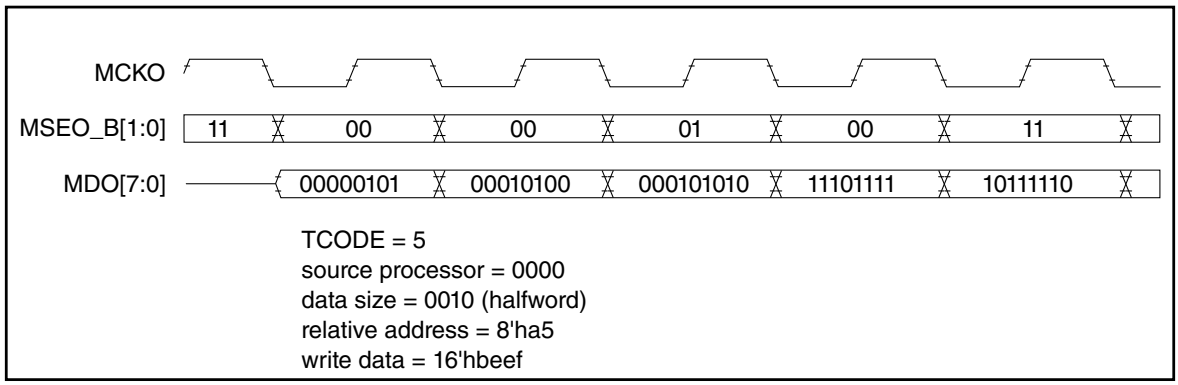


Figure 17-48. Data Trace - Data Write Message

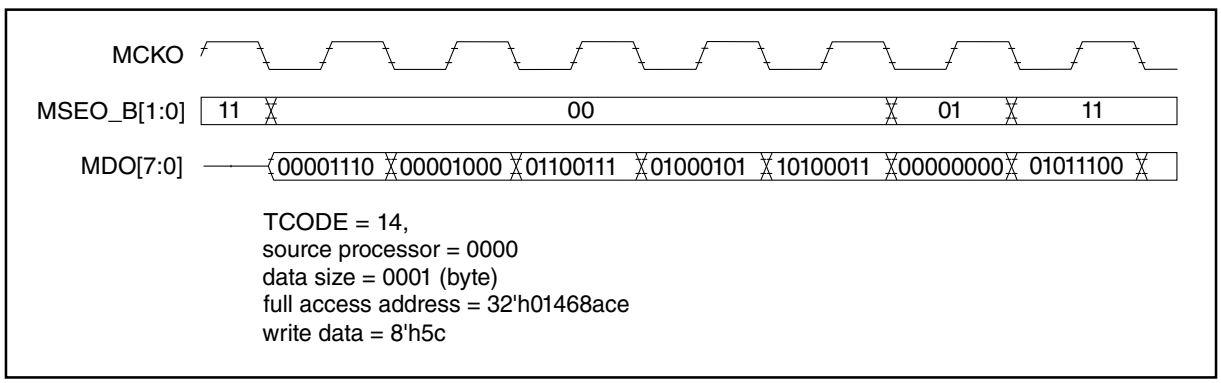


Figure 17-49. Data Trace — Data Read w/ Sync message

17.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

17.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

17.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

17.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z7260n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

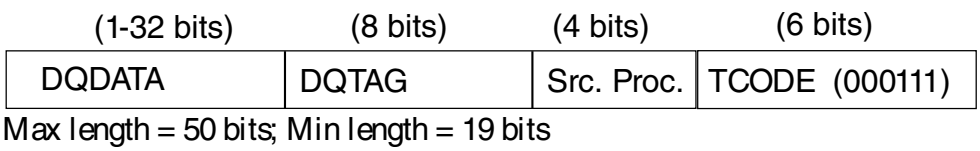


Figure 17-50. Data Acquisition message format

17.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z7260n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint

occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 17-39 for details on **evto_b**.

Watchpoint information is messaged out in the following format:

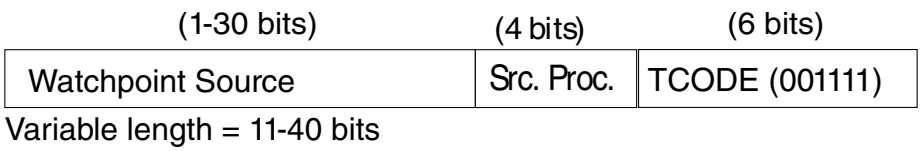


Figure 17-51. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 17-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
00000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX - Watchpoint #3 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX - Watchpoint #4 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX - Watchpoint #5 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX - Watchpoint #6 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX - Watchpoint #7 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX - Watchpoint #8 enabled for WTM	
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX - Watchpoint #9 enabled for WTM	
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX - Watchpoint #10 enabled for WTM	
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX - Watchpoint #11 enabled for WTM	
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX - Watchpoint #12 enabled for WTM	
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX - Watchpoint #13 enabled for WTM	
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX - Watchpoint #14 enabled for WTM	
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX - Watchpoint #15 enabled for WTM	
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX - Watchpoint #16 enabled for WTM	
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX - Watchpoint #17 enabled for WTM	
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX - Watchpoint #18 enabled for WTM	
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX - Watchpoint #19 enabled for WTM	
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX - Watchpoint #20 enabled for WTM	

Table 17-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #29 enabled for WTM

17.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

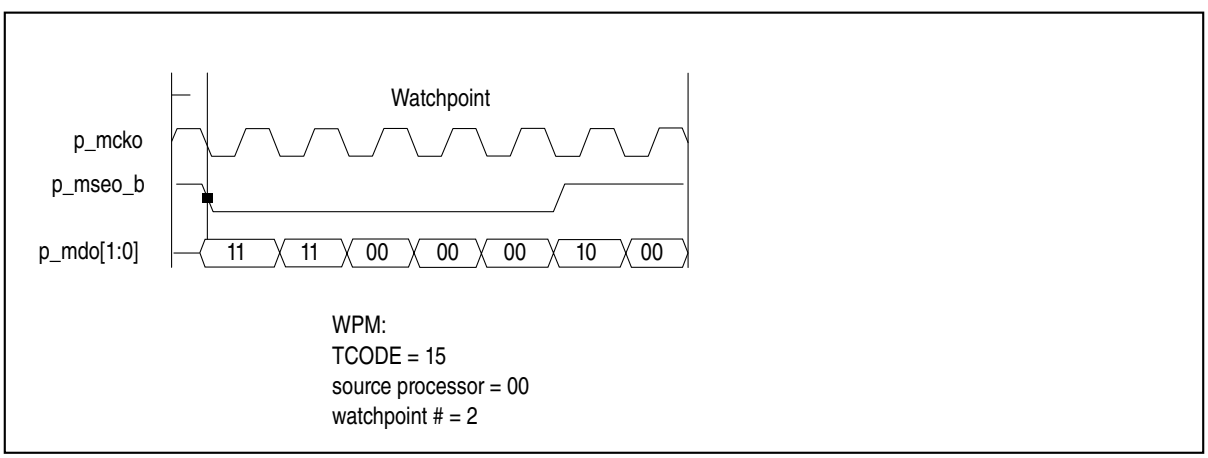


Figure 17-52. Watchpoint Message & Watchpoint Error message

17.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

17.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 17-39 for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

17.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and

the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

17.16.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 17-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

17.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 17-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the

RWCS Register, and asserts the **nex_rdy_b** pin (see Table 17-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

17.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

17.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set

3. The Error Message is sent (TCODE = 8) indicating Read/Write error

17.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

17.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 17-53. Error message format1

17.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

17.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 17-39. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z7260n3) Core Debug Support chapter.

Table 17-38. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 17-39.

Table 17-39. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.

Table continues on the next page...

Table 17-39. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted
evto_b	O	Event Out (**evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 17-40. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 17-42.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

17.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 17-41. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 17-41. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 17-54 illustrates the state diagram for single pin MSEO transfers.

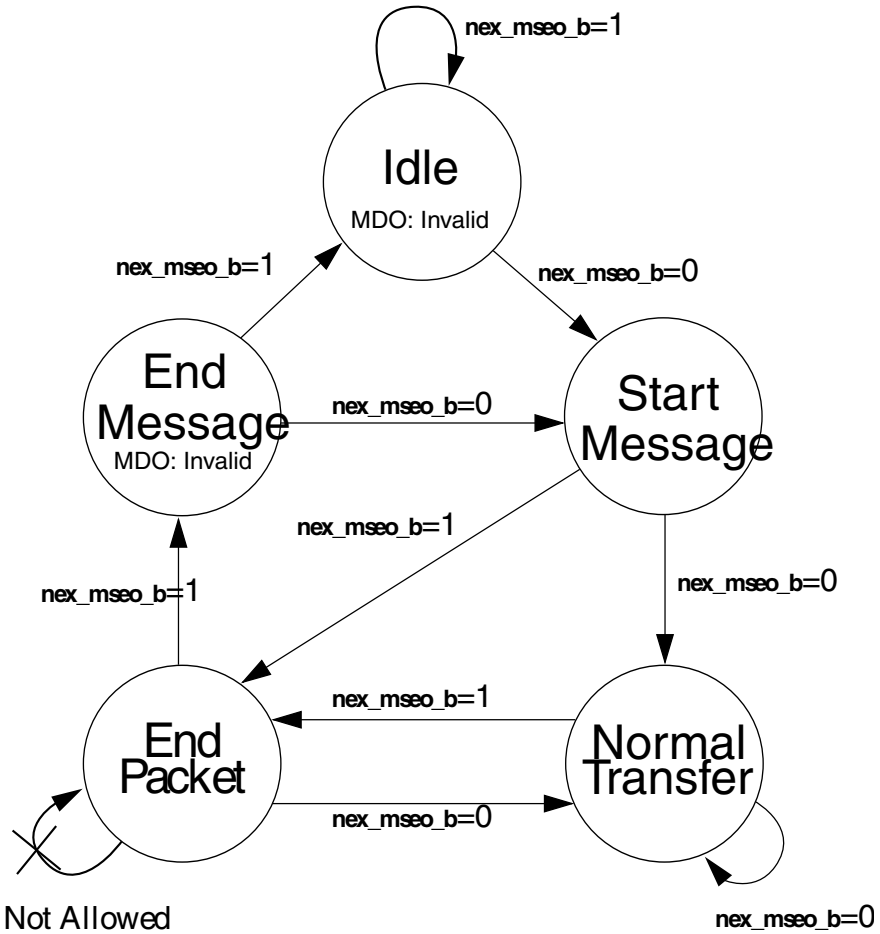


Figure 17-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 17-55 illustrates the state diagram for dual pin MSEO transfers.

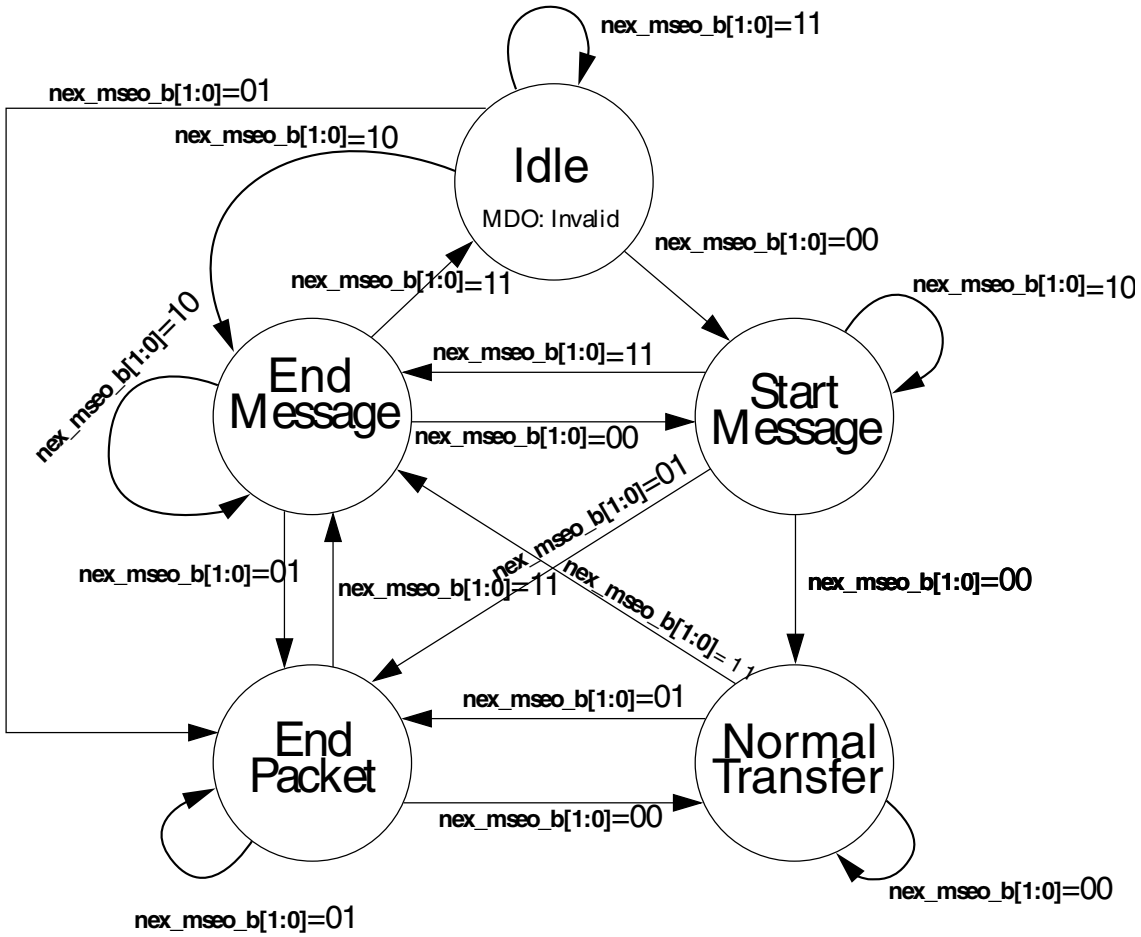


Figure 17-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

17.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

17.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 17-42 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 17-42. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

17.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 17-43 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 17-44 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 17-43. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 17-44. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 17-44. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 17-45 and Table 17-46 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 17-45. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 17-46. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 17-47 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 17-48 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 17-47. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 17-48. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

17.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

17.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

17.22.1 JTAG sequence for accessing internal Nexus registers

Table 17-49. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

17.22.2 JTAG sequence for read access of memory-mapped resources

Table 17-50. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

17.22.3 JTAG sequence for write access of memory-mapped resources

Table 17-51. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

Chapter 18
e200z4201n3 Core Debug Support

18.1 Chip-specific Core Debug support information

18.1.1 Debug Request During Waiting, Halted or Stopped State

When the CPU is in Waiting, Halted, or Stopped state and a debug request is asserted, the CPU asserts the p_wakeup signal to chip-level clock generator module, MC_CGM, so as to re-enable the m_clk input of the CPU. But the p_wakeup signal is not used in this device, so the CPU will not enter the debug mode via this mechanism when it is in Stopped, Halted, or Waiting State.

18.1.2 Breakpoint propagation

This device allows a watchpoint or debug request in any one core to cause a debug request to be propagated to any of the cores which have their external debug request input enabled in the Debug Control Register 0 (DBCR0[DEVT1] or DBCR0[DEVT2]).

The figure below shows the block diagram for breakpoint propagation. Here, the EVTO output from a core is connected to the DEVT1 or DEVT2 input to another core.

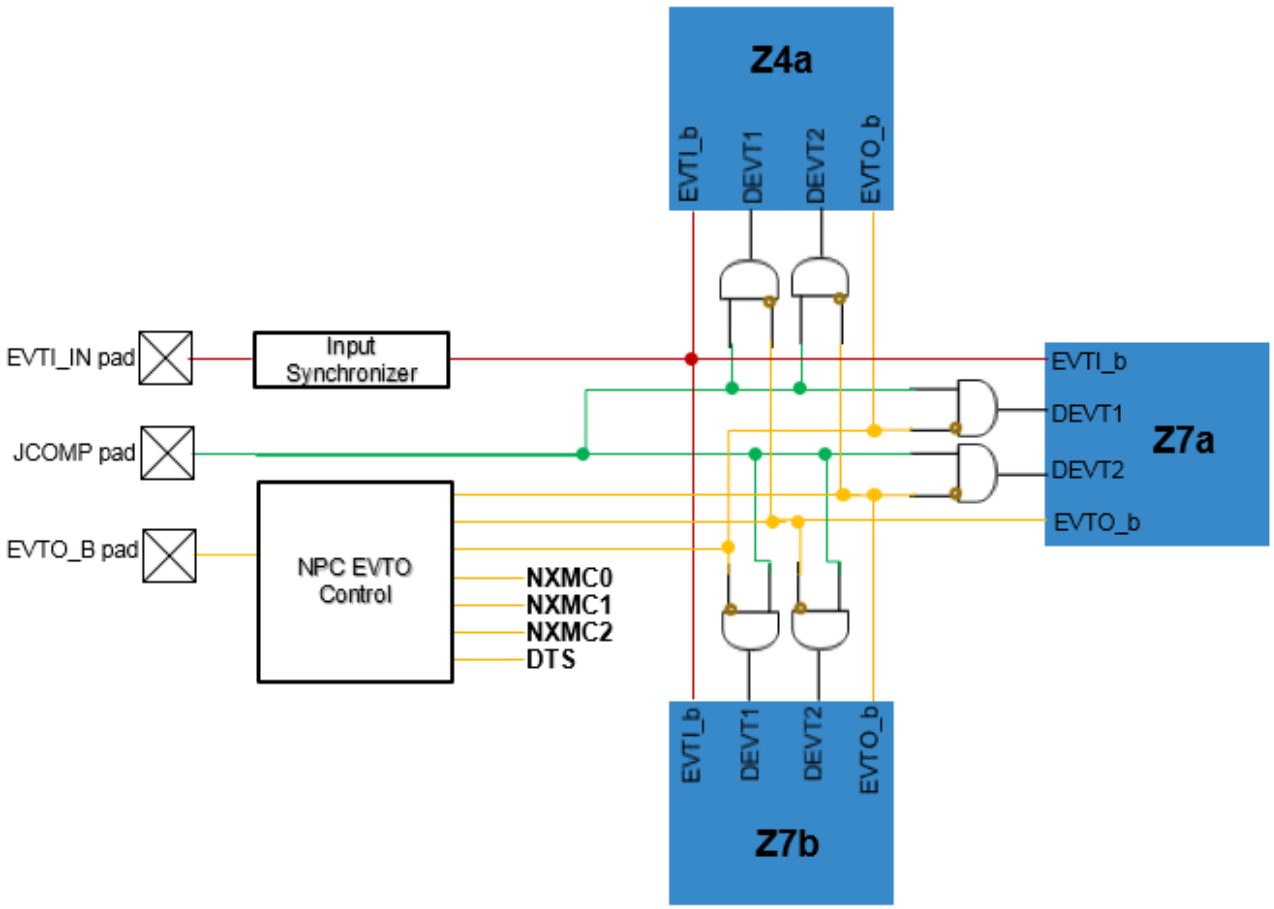


Figure 18-1. Breakpoint Propagation Connections

18.2 Overview

Internal debug support in the e200z4201n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *Power ISA 2.06*. e200z4201n3 supports a subset of these defined facilities. In addition to the facilities defined in *Power ISA 2.06*, e200z4201n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

18.2.1 Software Debug Facilities

e200z4201n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

18.2.1.1 *Power ISA 2.06* Compatibility

The e200z4201n3 core implements a subset of the *Power ISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

18.2.2 Additional Debug Facilities

In addition to the debug functionality defined in *Power ISA 2.06*, e200z4201n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

18.2.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z4201n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z4201n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

18.2.3 Hardware Debug Facilities

The e200z4201n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard IEEE 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z4201n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources

are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

18.2.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

18.2.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z4201n3 debug resources.

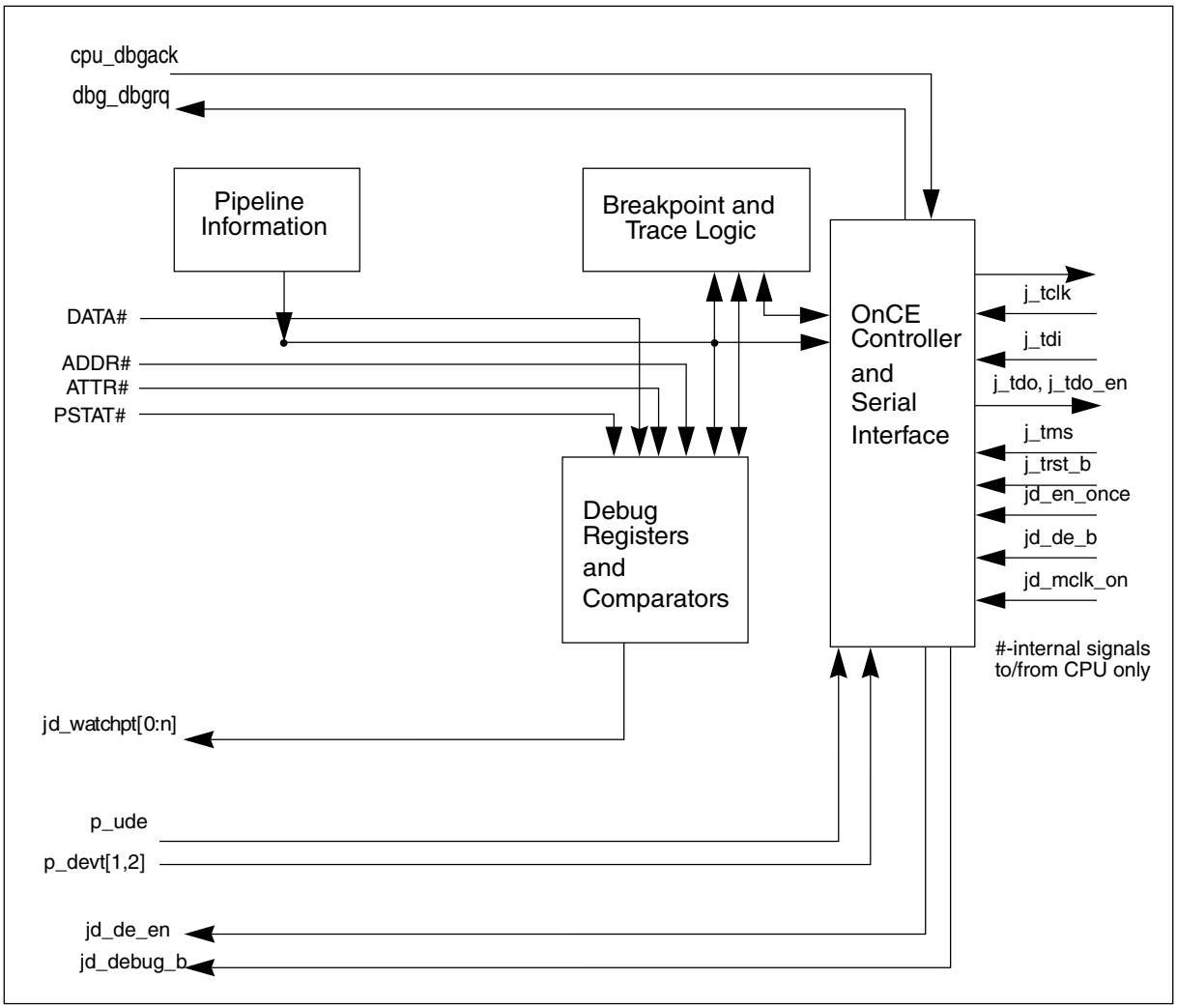


Figure 18-2. e200z4201n3 Debug Resources

18.3 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *Power ISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z4201n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z4201n3 debug configuration supports most of these event types. Unsupported *Power ISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

18.3.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

18.3.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *Power ISA 2.06* definition, Data Address Compare events on e200z4201n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC\_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/ EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

18.3.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/ store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC\_OFST}$ field when the DVC DAC status is recorded.

Table 18-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to

this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in Table 18-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 18-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table continues on the next page...

Table 18-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 18-2 through Table 18-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 18-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 18-2. DAC events and Resultant Updates, Dual-issue case 1 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 18-3. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6).

Table continues on the next page...

Table 18-3. DAC events and Resultant Updates, Dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (3): DSI, with or without DAC	**Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACyInstruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 18-4. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) alu** **(4) alu** **(5) load/store** **(6) alu**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 18-5. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

18.3.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1

may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

18.3.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

18.3.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken

debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

18.3.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

18.3.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

18.3.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

18.3.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

18.3.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

18.3.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

18.3.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

18.3.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

18.4 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

18.4.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are

not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z4201n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

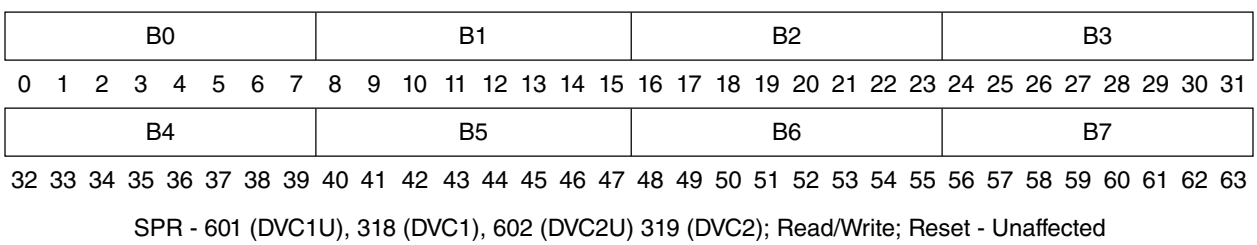


Figure 18-3. DVC1, DVC2 registers

18.4.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z4201n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

18.4.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z4201n3 adds some implementation specific bits to this register, as seen in the following figure.

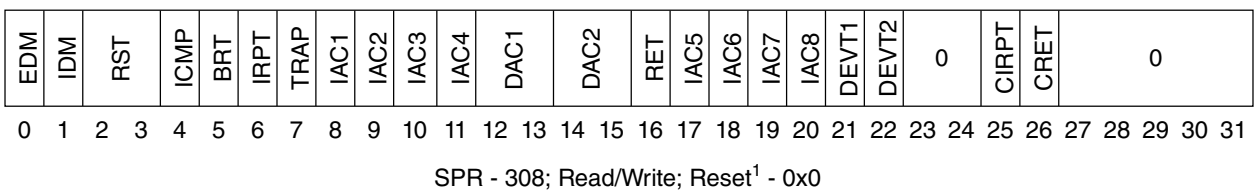


Figure 18-4. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 18-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled

Table continues on the next page...

Table 18-6. DBCR0 field descriptions (continued)

Bit	Name	Description
		1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled

Table continues on the next page...

Table 18-6. DBCR0 field descriptions (continued)

Bit	Name	Description
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

18.4.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.

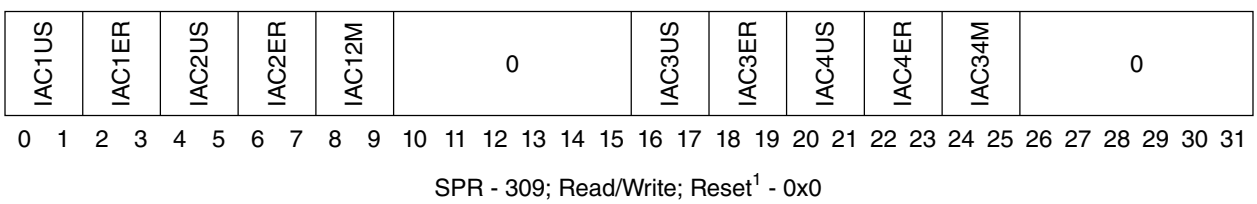


Figure 18-5. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 18-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode

Table continues on the next page...

Table 18-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 IAC1 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode
		00 IAC1 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode
		00 IAC2 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode
		00 IAC2 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode
		00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2.
		01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode
		00 IAC3 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode
		00 IAC3 debug events are based on effective address

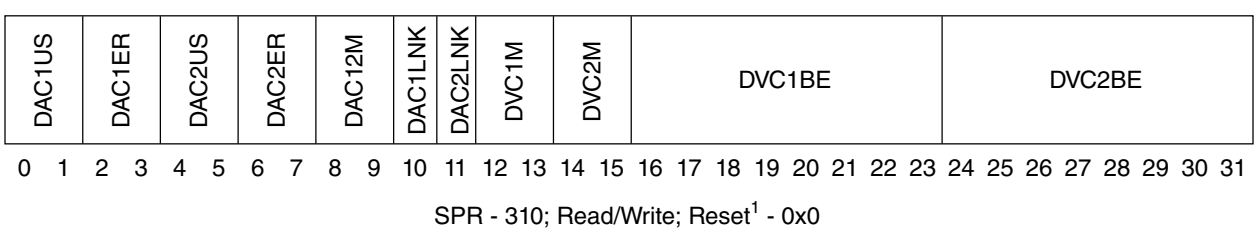
Table continues on the next page...

Table 18-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

18.4.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 18-6. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 18-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table continues on the next page...

Table 18-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access.

Table continues on the next page...

Table 18-8. DBCR2 Bit Definitions (continued)

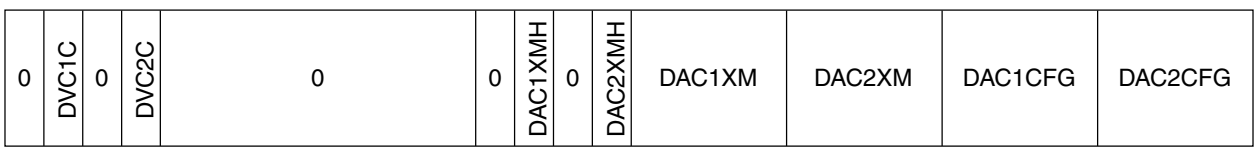
Bit	Name	Description
		10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2.

Table 18-8. DBCR2 Bit Definitions

Bit	Name	Description
		xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

18.4.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 18-7. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 18-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation.

Table continues on the next page...

Table 18-9. DBCR4 field description (continued)

Bit	Name	Description
		1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).

Table continues on the next page...

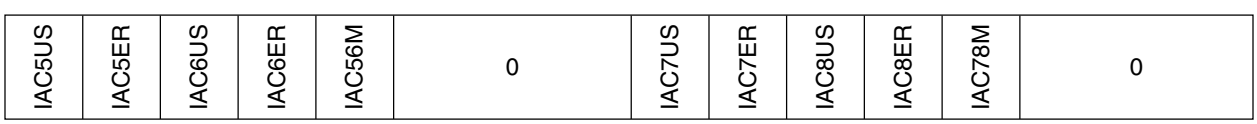
Table 18-9. DBCR4 field description (continued)

Bit	Name	Description
		NOTE: unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

18.4.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 18-8. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 18-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved

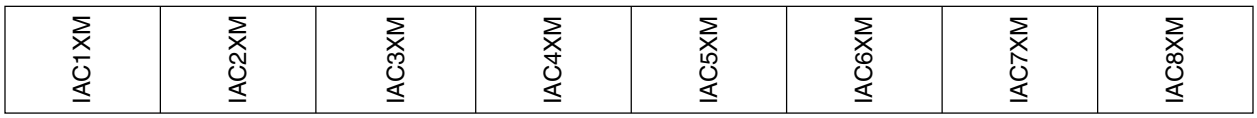
Table continues on the next page...

Table 18-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

18.4.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.



SPR - 603; Read/Write; Reset[1] - 0x0

Figure 18-9. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 18-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported.

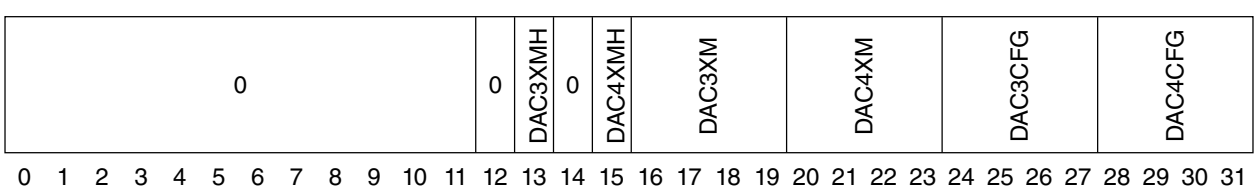
Table continues on the next page...

Table 18-11. DBCR6 field descriptions (continued)

Bit	Name	Description
		1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

18.4.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 18-10. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 18-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses

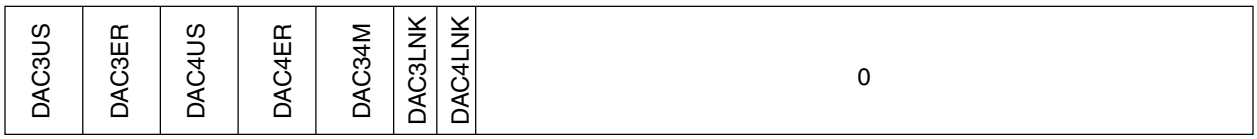
Table continues on the next page...

Table 18-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

18.4.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 18-11. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 18-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0

Table continues on the next page...

Table 18-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

18.4.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no

effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.

SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 18-12. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 18-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event

Table continues on the next page...

Table 18-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

18.4.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

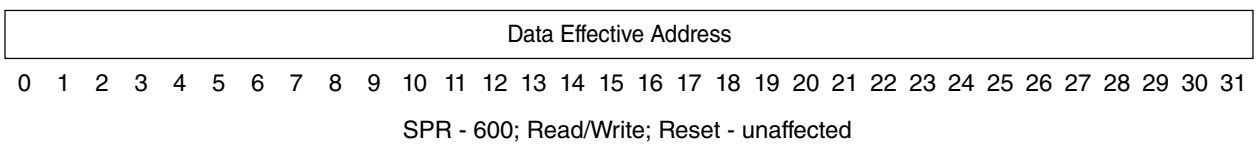


Figure 18-13. DDEAR Register

18.4.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

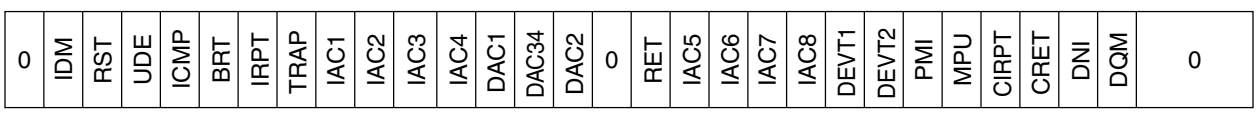
The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.



SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 18-14. EDBRAC0 Register

Table 18-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 18-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 18-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 18-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 18-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 18-16 shows which resources are controlled by EDBRAC0 settings.

Table 18-16. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$,

Table continues on the next page...

Table 18-16. EDBRAC0 Resource Control (continued)

$EDBRC0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												$DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table continues on the next page...

Table 18-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 18-16 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

18.4.4 Debug Event Select Register (DEVENT)

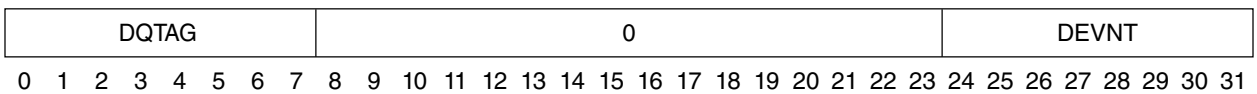
The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has

occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z4201n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.



SPR - 975; Reset[1] - 0x0

Figure 18-15. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 18-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock

Table 18-17. DEVENT Bit Definitions

Bit(s)	Name	Description
		x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

18.4.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z4201n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

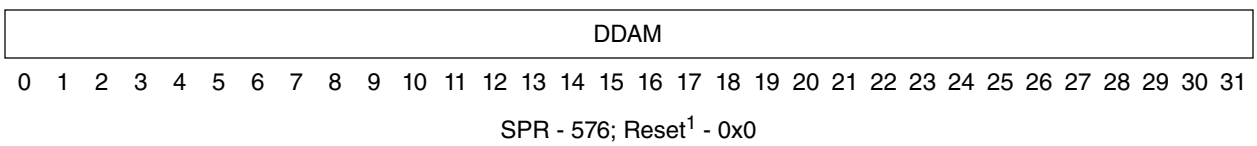


Figure 18-16. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 18-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

18.5 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 18-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0

pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 18-9 and Table 18-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

18.6 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is

performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

18.6.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

18.6.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

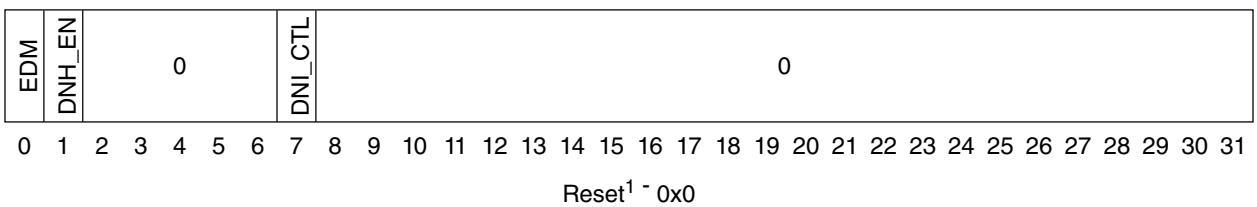


Figure 18-17. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

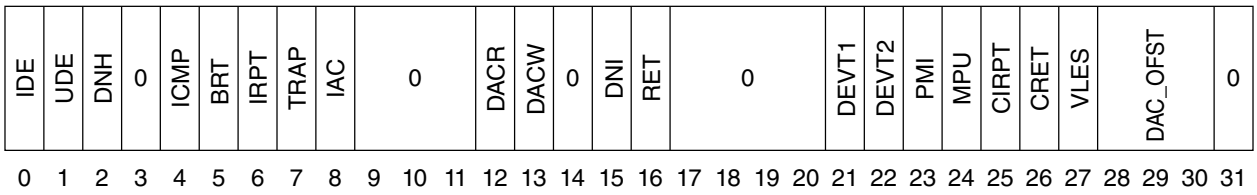
The following table provides bit definitions for External Debug Control Register 0.

Table 18-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

18.6.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.



Read/Write; Reset[1] - 0x0000_0000

Figure 18-18. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 18-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved

Table continues on the next page...

Table 18-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

18.6.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

Read/Write; Reset[1] - 0x0000_0000

Figure 18-19. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 18-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table continues on the next page...

Table 18-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

18.6.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR

hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

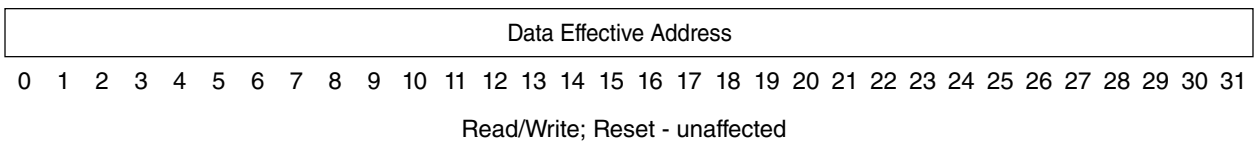


Figure 18-20. EDDEAR Register

18.6.2 OnCE Introduction

The e200z4201n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z4201n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/ software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

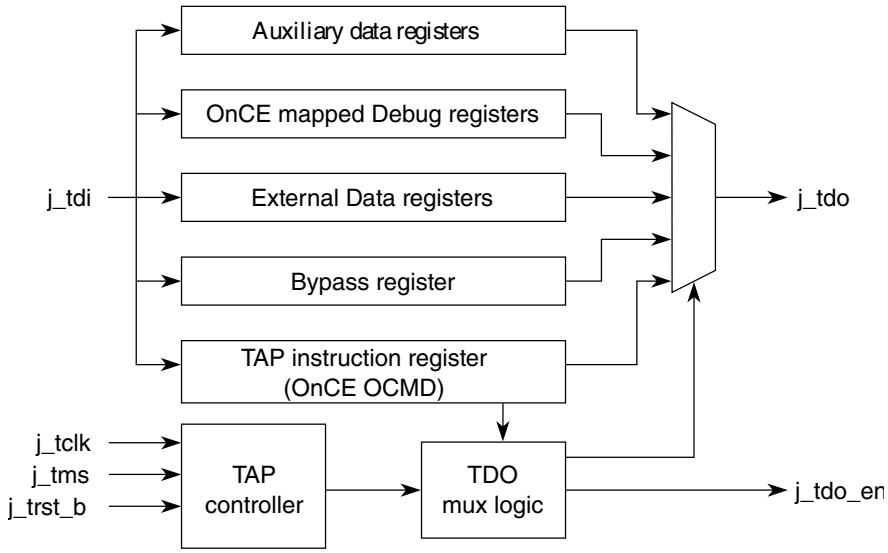
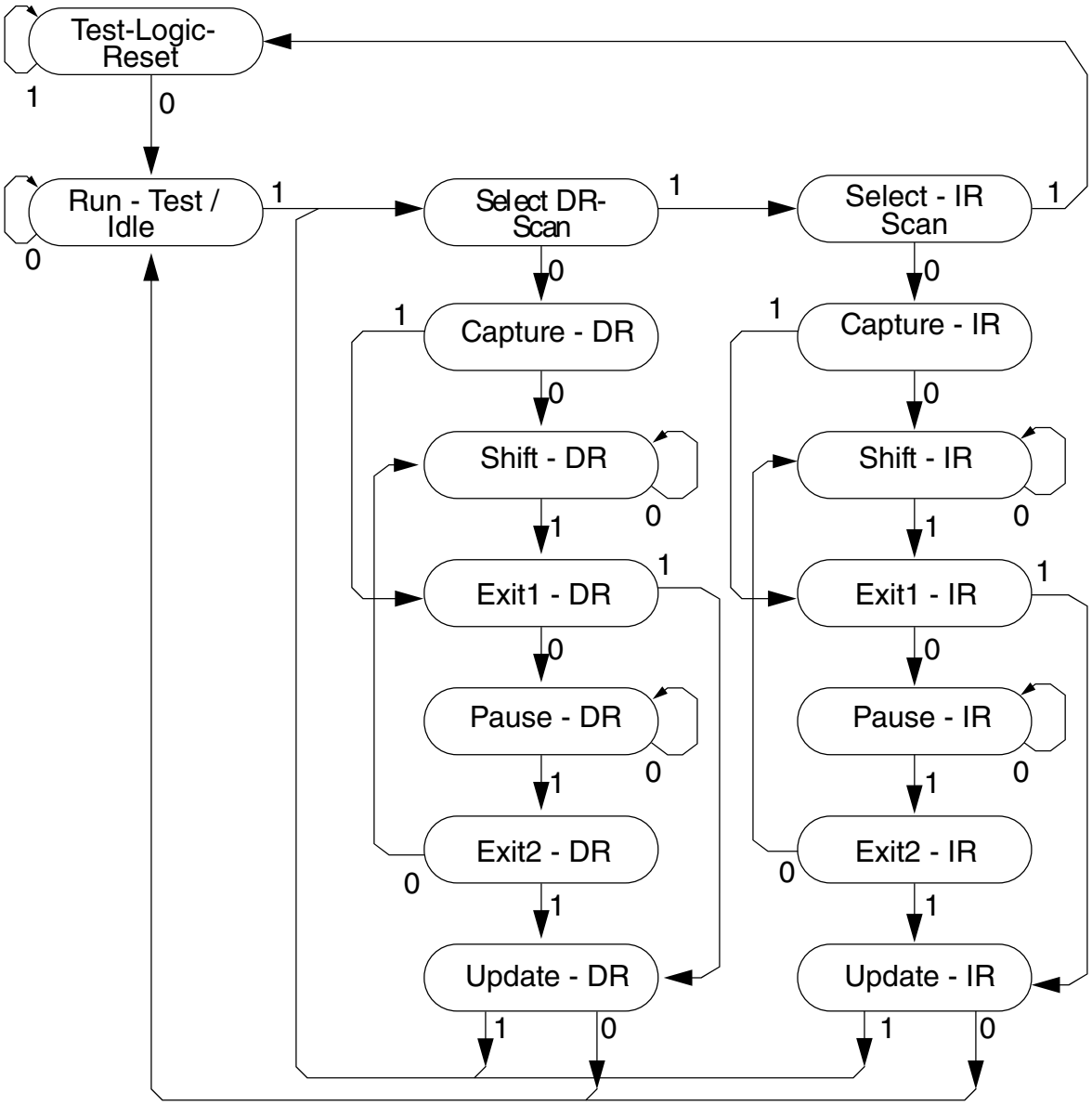


Figure 18-21. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

18.6.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 18-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z4201n3) Core Complex Overview.

18.6.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

18.6.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

18.6.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

18.6.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

18.6.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

18.6.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

18.6.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE

commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

18.6.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z4201n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

18.6.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

18.6.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z4201n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

18.6.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

18.6.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

18.6.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

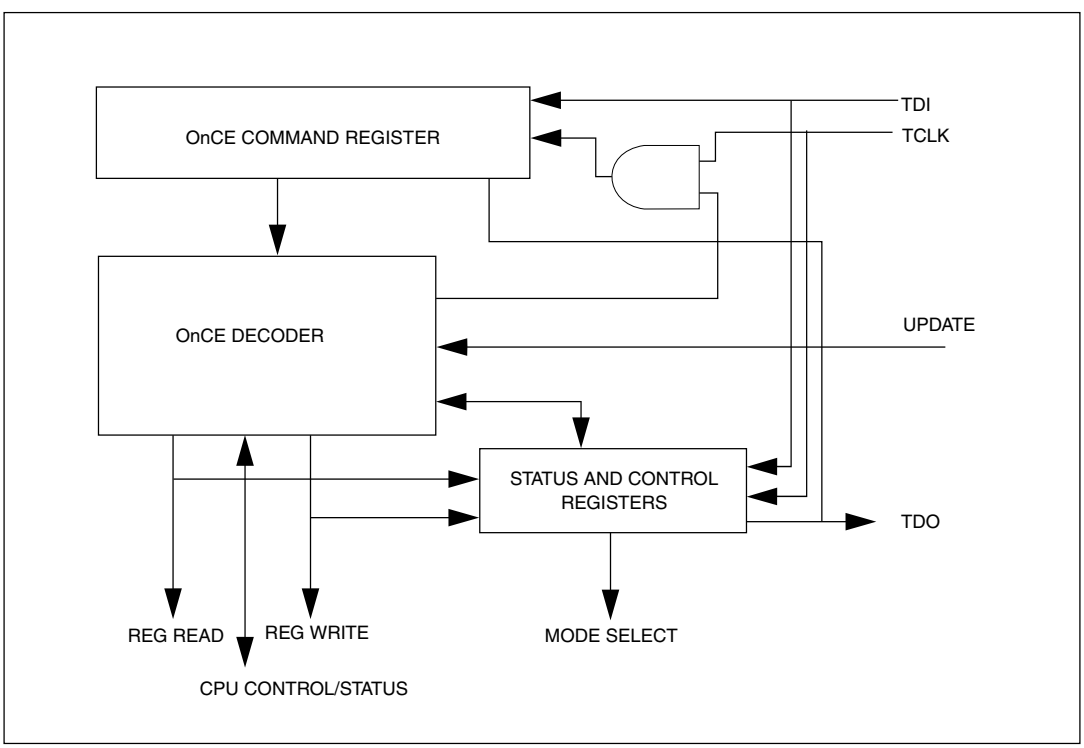


Figure 18-22. OnCE Controller and Serial Interface

18.6.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 18-23. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 18-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

Table 18-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

18.6.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 18-24. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 18-24. OnCE Command Register

Table 18-24 provides bit definitions for the Once Command Register.

Table 18-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6]

Table continues on the next page...

Table 18-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 18-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 18-25 indicates the OnCE register addresses.

Table 18-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved

Table continues on the next page...

Table 18-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)

Table continues on the next page...

Table 18-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

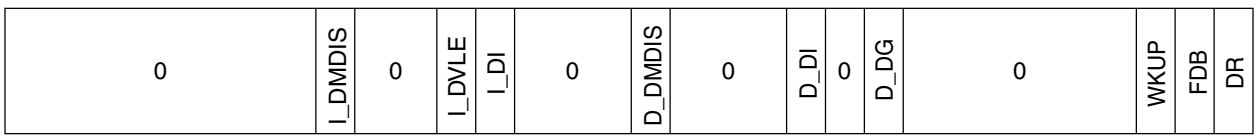
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

18.6.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z4201n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 18-25. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 18-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions

Table continues on the next page...

Table 18-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

18.6.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z4201n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z4201n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z4201n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 18-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1]	
DBCR0	Y	Y	Y	N	*[1]	*$EDBCR0_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	

Table continues on the next page...

Table 18-27. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

18.6.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

18.6.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z4201n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z4201n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

18.6.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

18.6.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

18.6.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled

along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

18.6.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z4201n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

18.6.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z4201n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

18.6.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

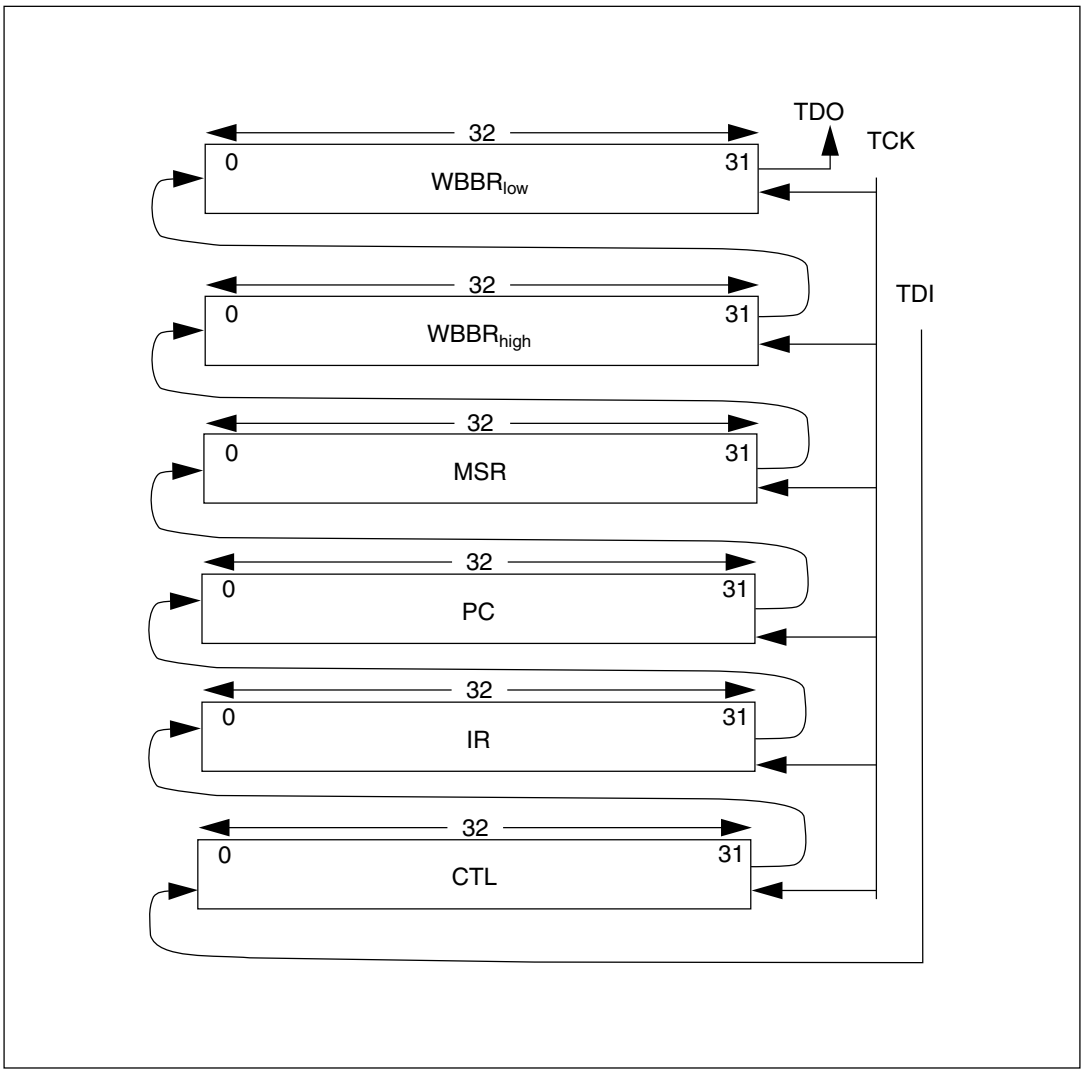


Figure 18-26. CPU Scan Chain Register (CPUSCR)

18.6.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

18.6.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

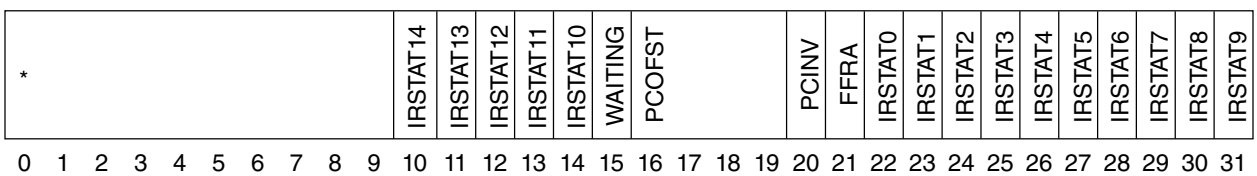


Figure 18-27. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache .

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the Power ISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a Power ISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

18.6.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

18.6.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

18.6.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

18.6.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

18.6.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

18.7 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

18.8 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

18.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode.
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the CPU has re-entered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should re-write the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 19
z7 Core Debug Support

19.1 Overview

Internal debug support in the e200z7260n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of Power ISA 2.06. e200z7260n3 supports a subset of these defined facilities. In addition to the facilities defined in Power ISA 2.06, e200z7260n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

19.1.1 Software Debug Facilities

e200z7260n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

19.1.1.1 Power ISA 2.06 Compatibility

The e200z7260n3 core implements a subset of the Power ISA 2.06 internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

19.1.2 Additional Debug Facilities

In addition to the debug functionality defined in Power ISA 2.06, e200z7260n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

19.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z7260n3 core. This port provides information corresponding to each read or write access

completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z7260n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

19.1.3 Hardware Debug Facilities

The e200z7260n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z7260n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

19.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

19.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z7260n3 debug resources.

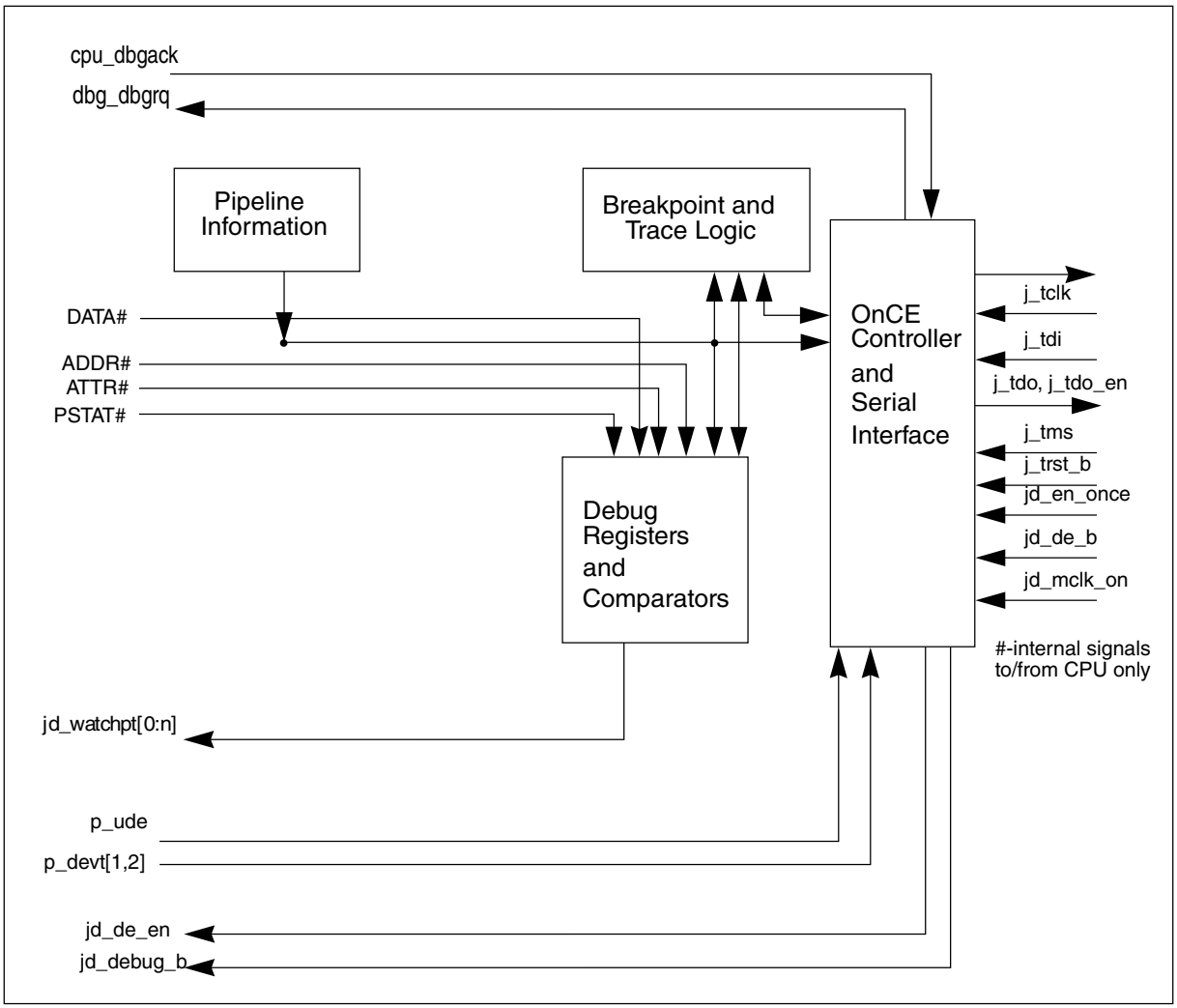


Figure 19-1. e200z7260n3 Debug Resources

19.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by Power ISA 2.06:

- Instruction Address Compare debug events
- Data Address Compare debug events
- Trap debug events
- Branch Taken debug events
- Instruction Complete debug events
- Interrupt/Critical Interrupt Taken debug events
- Return/Critical Return debug events
- Unconditional debug events

In addition, e200z7260n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z7260n3 debug configuration supports most of these event types. Unsupported Power ISA 2.06 defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

19.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

19.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the Power ISA 2.06 definition, Data Address Compare events on e200z7260n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC\_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/ EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

19.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/ store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC\_OFST}$ field when the DVC DAC status is recorded.

Table 19-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to

this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in Table 19-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 19-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table continues on the next page...

Table 19-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 19-2 through Table 19-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 19-2. DAC events and Resultant Updates, dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 19-2. DAC events and Resultant Updates, dual-issue case 1 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 19-3. DAC events and Resultant Updates, dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6).

Table continues on the next page...

Table 19-3. DAC events and Resultant Updates, dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (3): DSI, with or without DAC	**Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACyInstruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 19-4. DAC events and Resultant Updates, dual-issue case 3

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) alu** **(4) alu** **(5) load/store** **(6) alu**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 19-5. DAC events and Resultant Updates, dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

19.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1

may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

19.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

19.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken

debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

19.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

19.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

19.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

19.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

19.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

19.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

19.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

19.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

19.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

19.3.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are

not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z7260n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

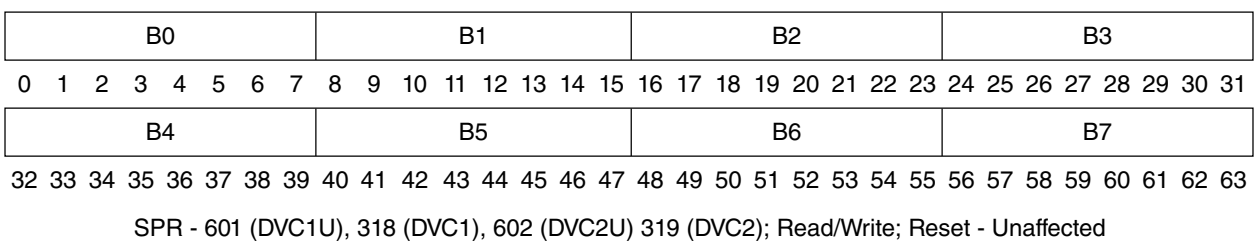


Figure 19-2. DVC1, DVC2 registers

19.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z7260n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

19.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z7260n3 adds some implementation specific bits to this register, as seen in the following figure.

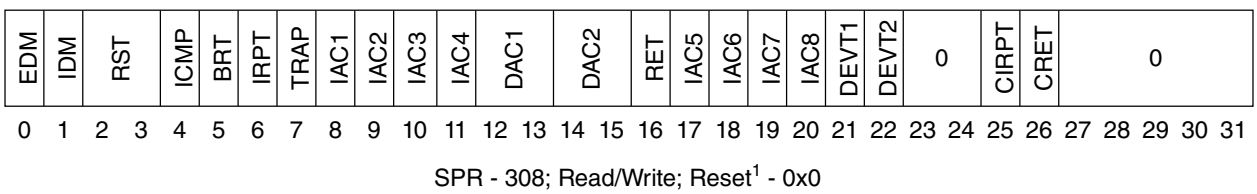


Figure 19-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 19-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled

Table continues on the next page...

Table 19-6. DBCR0 field descriptions (continued)

Bit	Name	Description
		1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled

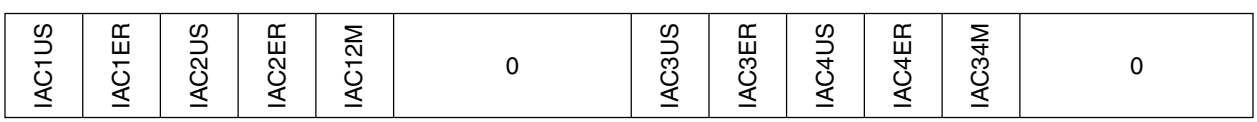
Table continues on the next page...

Table 19-6. DBCR0 field descriptions (continued)

Bit	Name	Description
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

19.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.



SPR - 309; Read/Write; Reset[1] - 0x0

Figure 19-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 19-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode

Table continues on the next page...

Table 19-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 IAC1 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode
		00 IAC1 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode
		00 IAC2 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode
		00 IAC2 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode
		00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2.
		01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode
		00 IAC3 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode
		00 IAC3 debug events are based on effective address

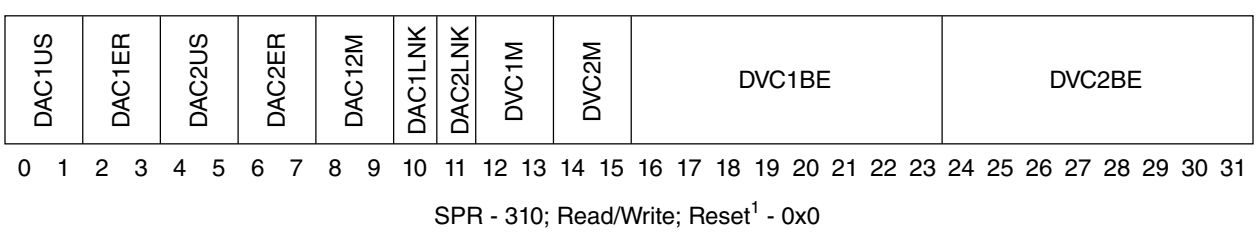
Table continues on the next page...

Table 19-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

19.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 19-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 19-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table continues on the next page...

Table 19-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access.

Table continues on the next page...

Table 19-8. DBCR2 Bit Definitions (continued)

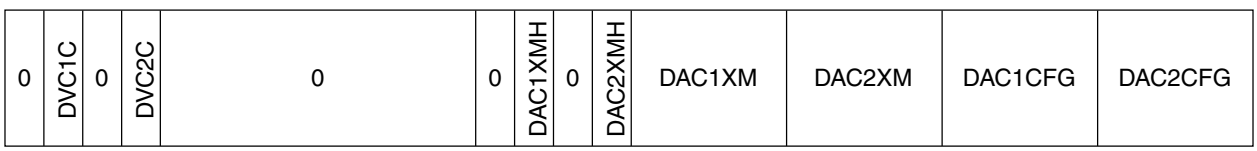
Bit	Name	Description
		10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2.

Table 19-8. DBCR2 Bit Definitions

Bit	Name	Description
		xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

19.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 19-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 19-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation.

Table continues on the next page...

Table 19-9. DBCR4 field description (continued)

Bit	Name	Description
		1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 - DAC1XM masks 0–15 low-order address bits 1 - DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).

Table continues on the next page...

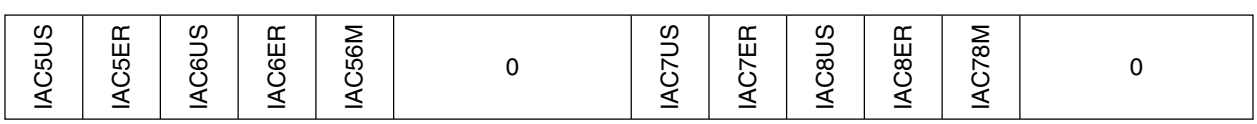
Table 19-9. DBCR4 field description (continued)

Bit	Name	Description
		NOTE: unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

19.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 19-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 19-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved

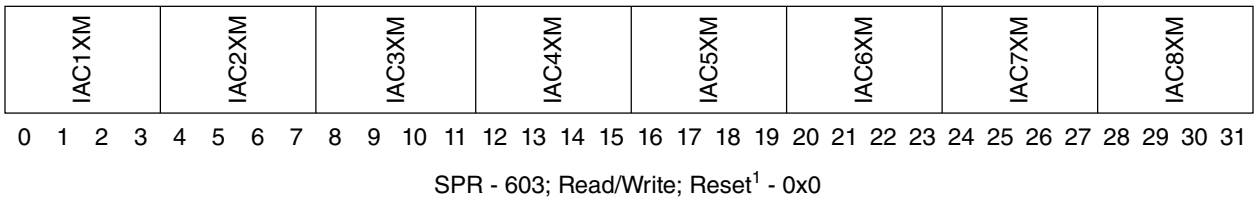
Table continues on the next page...

Table 19-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

19.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.



SPR - 603; Read/Write; Reset[1] - 0x0

Figure 19-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 19-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported.

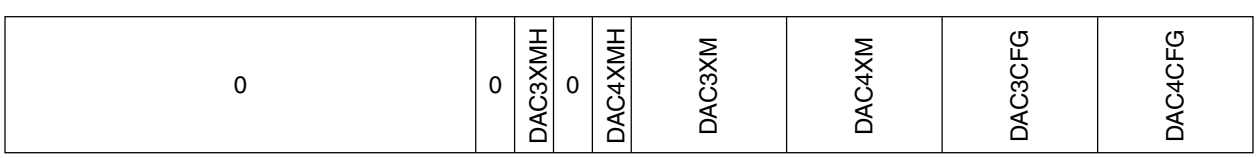
Table continues on the next page...

Table 19-11. DBCR6 field descriptions (continued)

Bit	Name	Description
		1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

19.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 19-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 19-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses

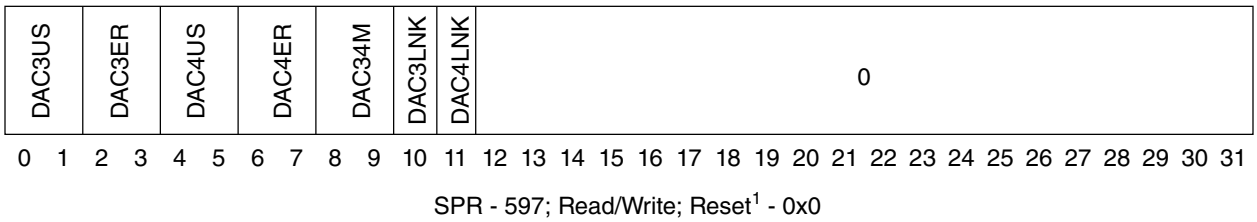
Table continues on the next page...

Table 19-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

19.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 19-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 19-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0

Table continues on the next page...

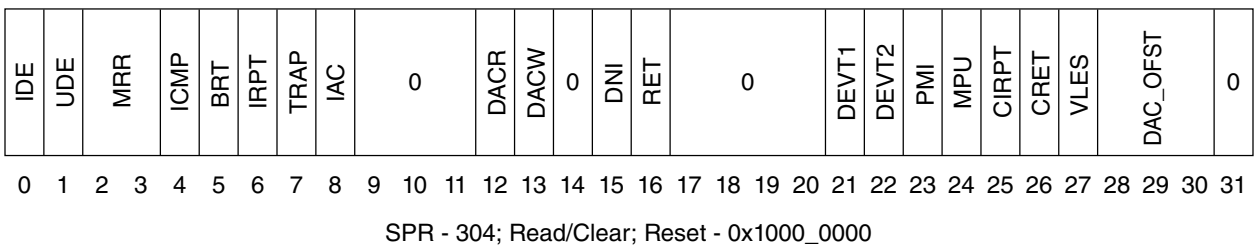
Table 19-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

19.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no

effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.



SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 19-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 19-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event

Table continues on the next page...

Table 19-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

19.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

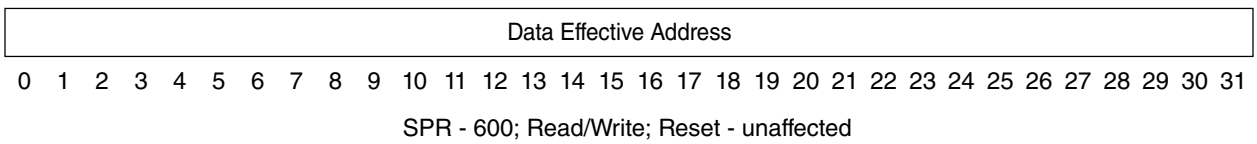


Figure 19-12. DDEAR Register

19.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

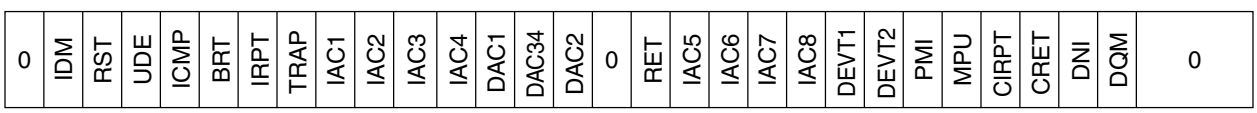
The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.



SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 19-13. EDBRAC0 Register

Table 19-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 19-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 19-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 19-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 19-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 19-16 shows which resources are controlled by EDBRAC0 settings.

Table 19-16. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$,

Table continues on the next page...

Table 19-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												$DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table continues on the next page...

Table 19-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 19-16 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

19.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has

occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z7260n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.

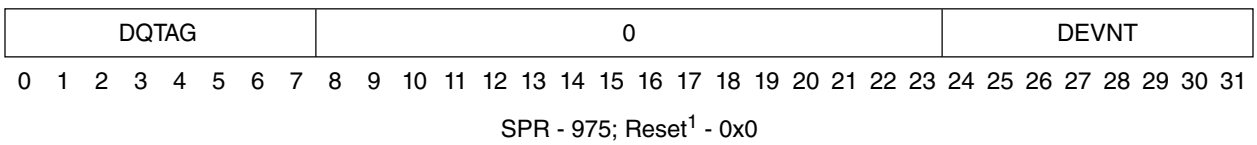


Figure 19-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 19-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - p_devnt_out[0] and jd_watchpt[12] are asserted for one clock xxxxxx1x - p_devnt_out[1] and jd_watchpt[13] are asserted for one clock xxxxx1xx - p_devnt_out[2] and jd_watchpt[20] are asserted for one clock xxxx1xxx - p_devnt_out[3] and jd_watchpt[21] are asserted for one clock xxx1xxxx - p_devnt_out[4] is asserted for one cloc kxx1xxxxx - p_devnt_out[5] is asserted for one clock

Table 19-17. DEVENT Bit Definitions

Bit(s)	Name	Description
		x1xxxxxx - p_devnt_out[6] is asserted for one clock 1xxxxxxx - p_devnt_out[7] is asserted for one clock

19.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z7260n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

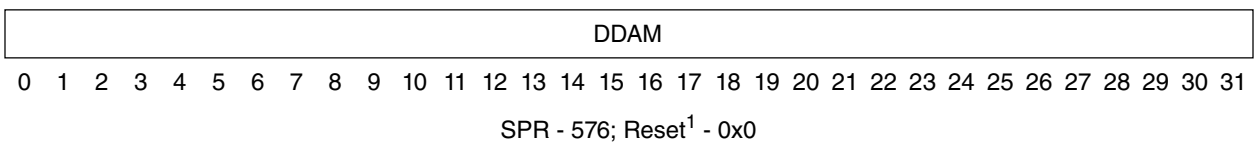


Figure 19-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 19-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

19.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 19-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0

pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 19-9 and Table 19-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

19.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is

performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

19.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

19.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

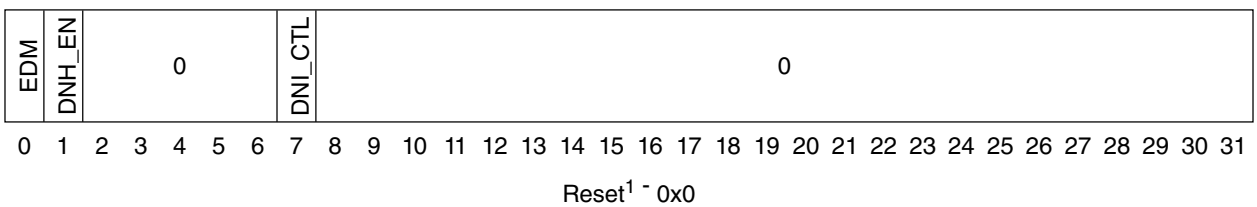


Figure 19-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 19-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

19.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

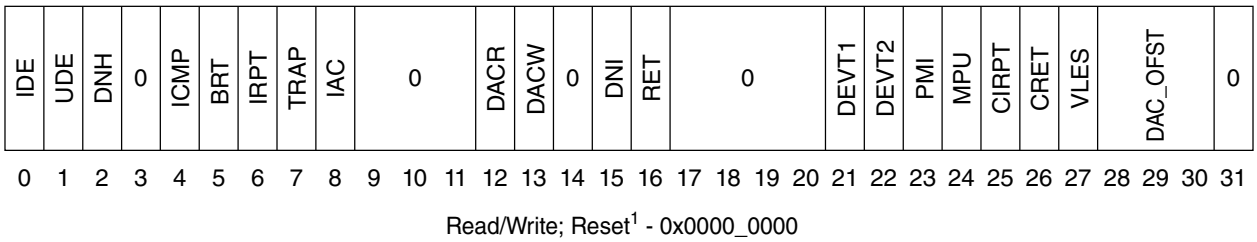


Figure 19-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 19-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved

Table continues on the next page...

Table 19-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

19.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

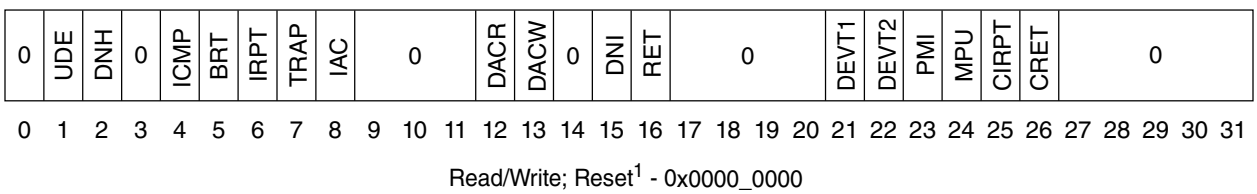


Figure 19-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 19-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table continues on the next page...

Table 19-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

19.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR

hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

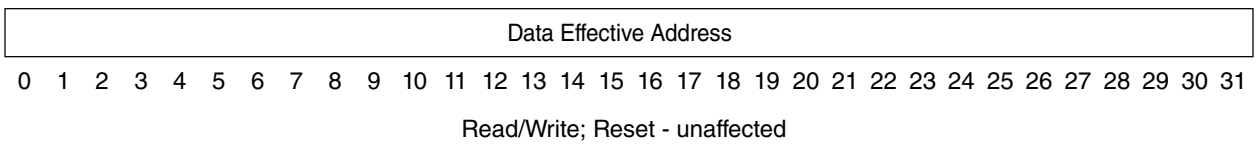


Figure 19-19. EDDEAR Register

19.5.2 OnCE Introduction

The e200z7260n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z7260n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/ software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

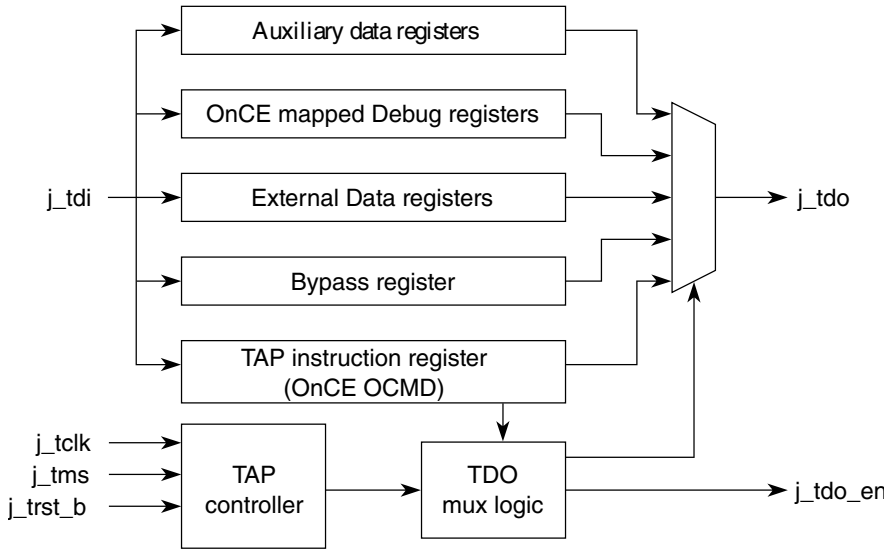
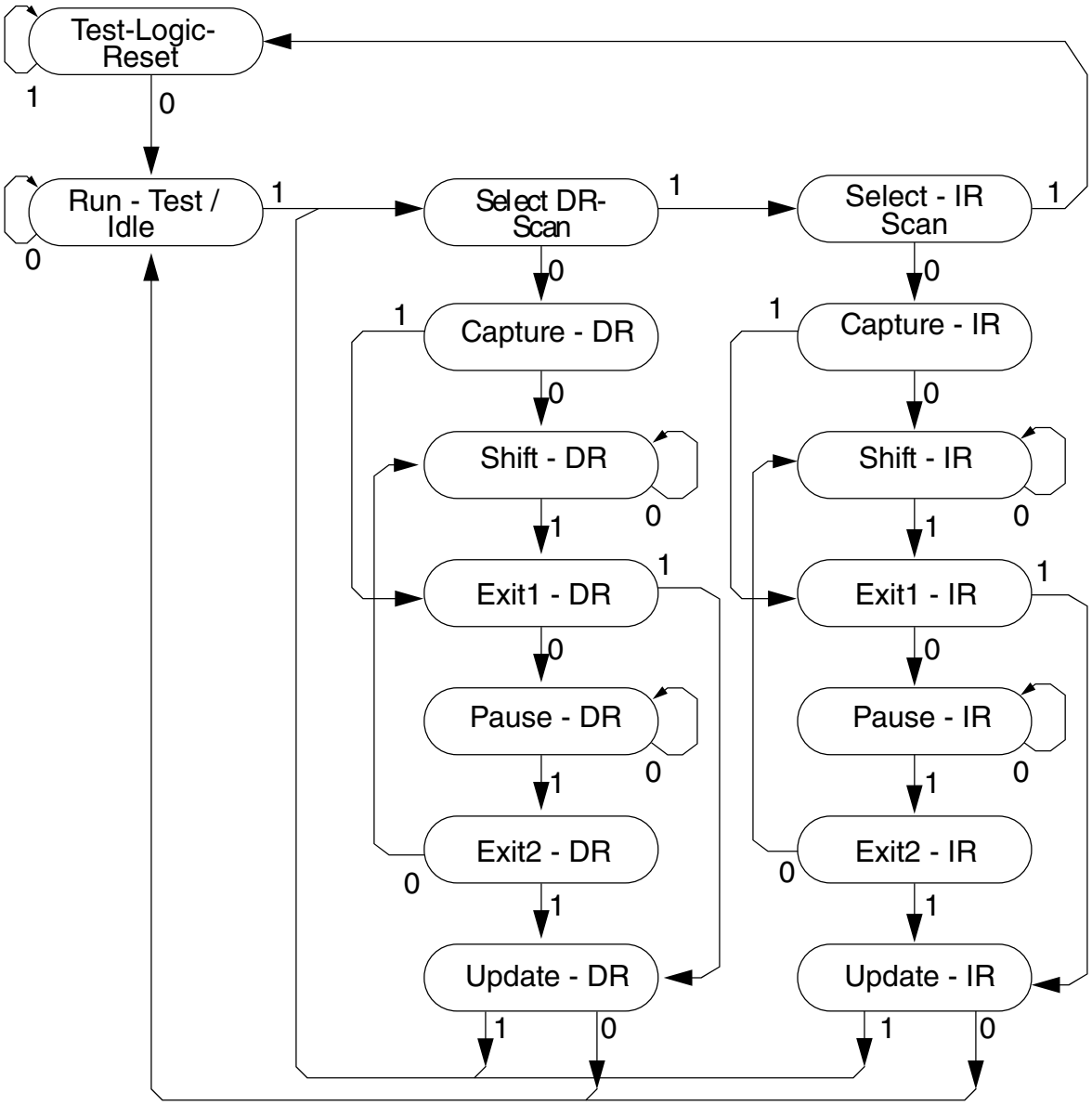


Figure 19-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

19.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 19-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z7260n3) Core Complex Overview.

19.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

19.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

19.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

19.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

19.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

19.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

19.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE

commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

19.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z7260n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

19.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

19.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z7260n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

19.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

19.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

19.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in or parallel loading commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

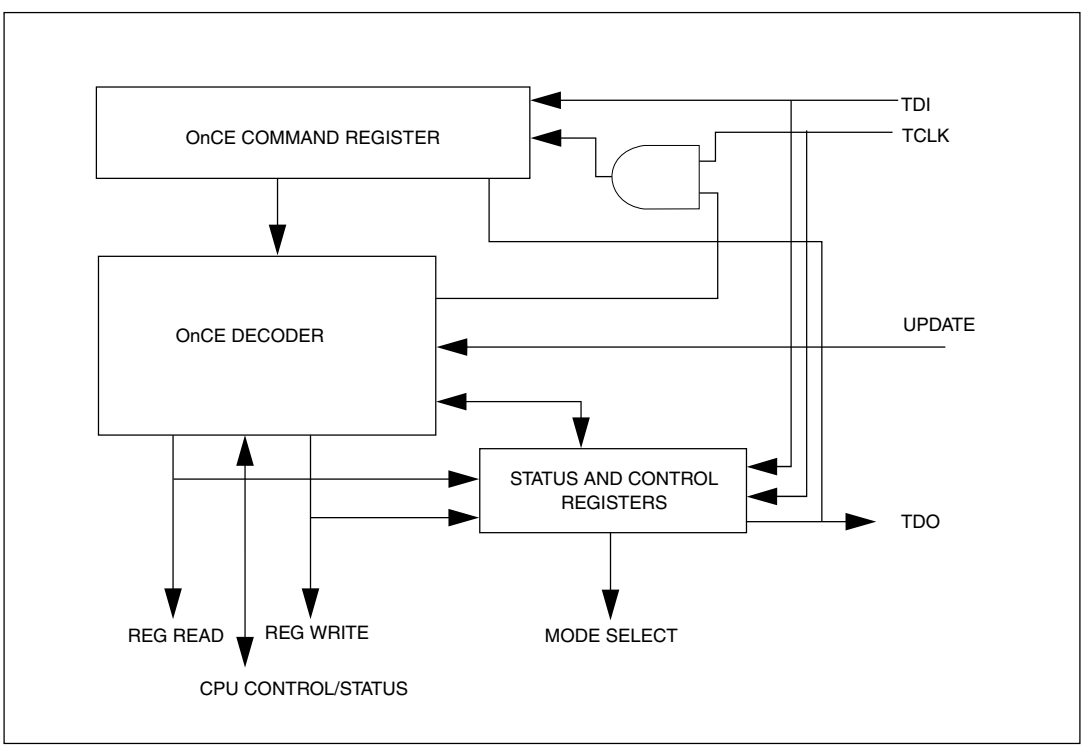


Figure 19-21. OnCE Controller and Serial Interface

19.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 19-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 19-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

Table 19-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

19.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 19-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 19-23. OnCE Command Register

Table 19-24 provides bit definitions for the Once Command Register.

Table 19-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6]

Table continues on the next page...

Table 19-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 19-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 19-25 indicates the OnCE register addresses.

Table 19-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved

Table continues on the next page...

Table 19-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)

Table continues on the next page...

Table 19-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

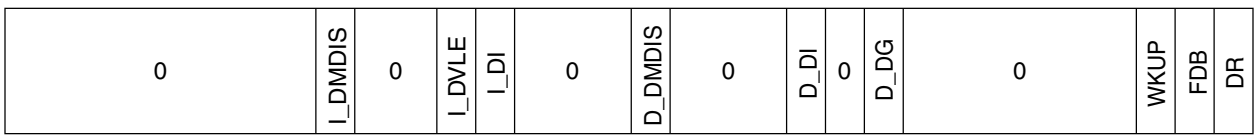
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

19.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z7260n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 19-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 19-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions

Table continues on the next page...

Table 19-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

19.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z7260n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z7260n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z7260n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 19-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1]	
DBCR0	Y	Y	Y	N	*[1]	*$EDBCR0_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	

Table continues on the next page...

Table 19-27. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

19.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

19.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z7260n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z7260n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

19.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

19.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

19.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled

along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

19.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z7260n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

19.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z7260n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

19.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

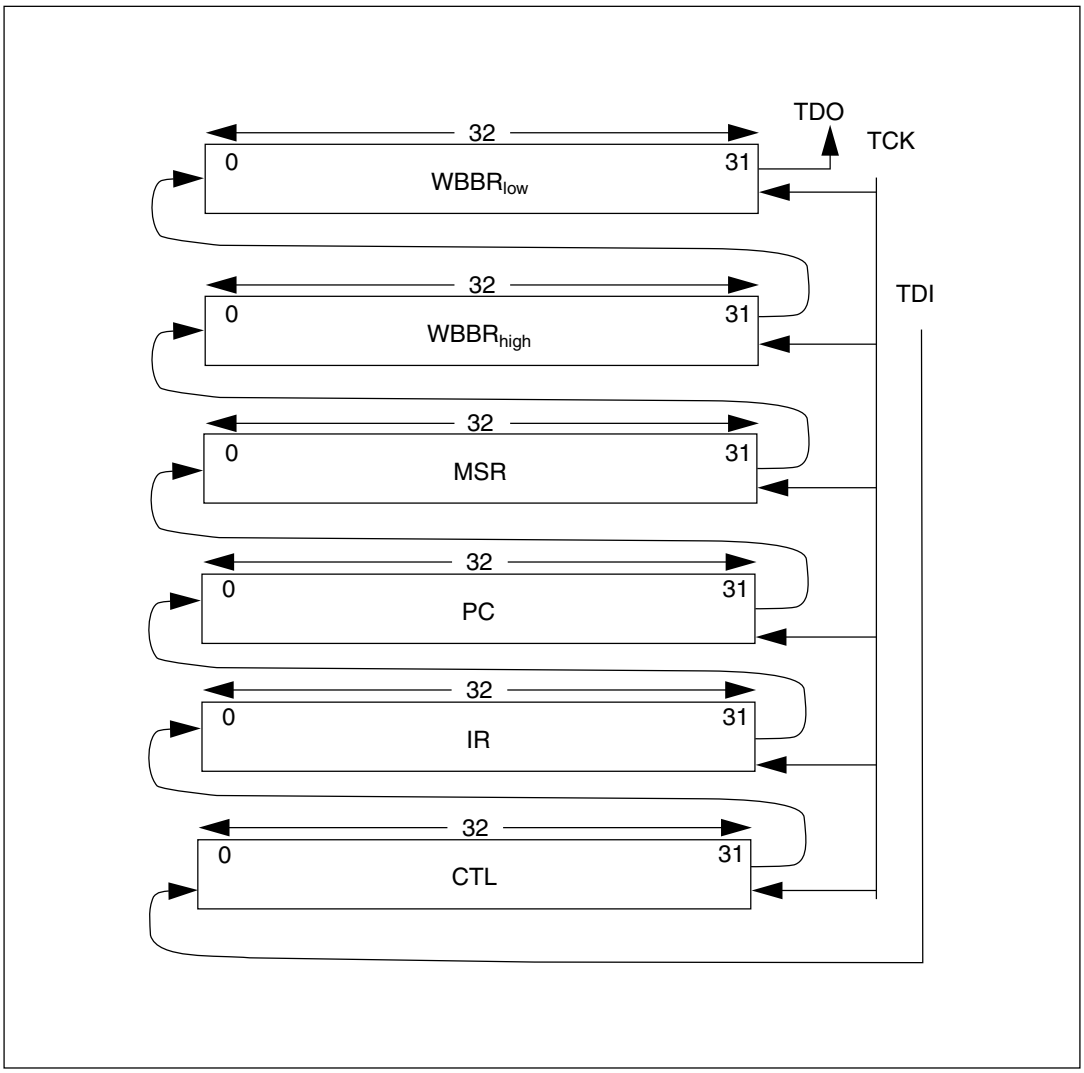


Figure 19-25. CPU Scan Chain Register (CPUSCR)

19.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

19.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

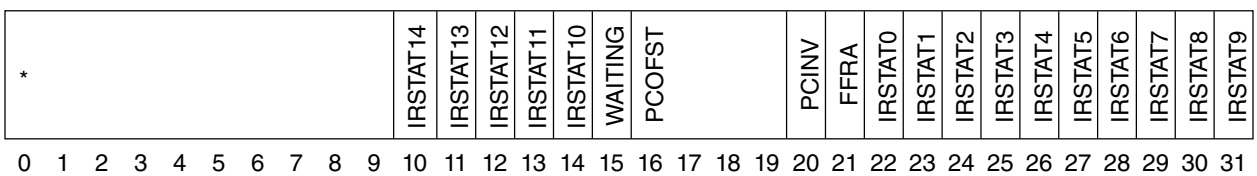


Figure 19-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache .

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the PowerISA VLE status for the IR.

0= IR contains a booke instruction. (unused encoding)

1= IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

19.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

19.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

19.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

19.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

19.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

19.6 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

19.7 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

19.8 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set
- Clear the DBSR status bits
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the CPU has re-entered the Debug state and that no ERR condition occurred

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit
- The debugger should write the DBSR to a cleared state
- The debugger should re-write the DBCR0 with all bits including EDM cleared
- The debugger should clear the OCR_{DR} bit
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 20
Signal Processing Extension (SPE)

20.1 Introduction

This chapter provides an overview of the signal processing engine, version 2.1, which is designed to accelerate signal processing applications normally suited to DSP operation. This is accomplished using short vectors (two, four, or eight elements) within 64-bit GPRs and using single instruction multiple data (SIMD) operations to perform the requisite computations. SPE2.1 also architects an accumulator register to allow for certain back to back operations without loop unrolling. SPE2.1 is fully backward compatible with the original SPE. The remainder of this document uses the term SPE to refer to version 2.1 unless otherwise noted.

20.2 Nomenclature and Conventions

Several conventions regarding nomenclature are used in this chapter:

- Due to historical precedent, the terms SPE and SIMD are sometimes used interchangeably.
- The signal processing engine is abbreviated as SPE.
- All register bit numbering is 64-bit with bit 0 being the most significant bit. Registers that are only 32-bit define bit 32 as the most significant bit. For both 32-bit and 64-bit registers, bit 63 is the least significant bit.
- Bits 0–31 of a 64-bit register are referenced as word 0, upper word, even word, or high word element of the register. Bits 32–63 are referred to as word 1, lower word, odd word, or low word element of the register. Each word is an element of a 64-bit GPR.

- Bits 0–15 of a 64-bit register are referenced as half word 0. Bits 16–31 are referred to as half word 1. Bits 32–47 are referenced as half word 2. Bits 48–63 are referred to as half word 3. Each half word is an element of a 64-bit GPR.
- Bits 0–7 of a 64-bit register are referenced as byte 0. Bits 8–15 are referred to as byte 1. Bits 16–23 are referenced as byte 2. Bits 24–31 are referred to as byte 3. Bits 32–39 are referred to as byte 4. Bits 40–47 are referenced as byte 5. Bits 48–55 are referred to as byte 6. Bits 56–63 are referenced as byte 7. Each byte is an element of a 64-bit GPR.
- Bits 0–15 and bits 32–47 are referenced as even half words. Bits 16–31 and bits 48–63 are referenced as odd half words. Bits 0–15 and bits 16–31 are referenced as upper half words. Bits 32–47 and bits 48–63 are referenced as lower half words.
- Mnemonics for SPE instructions generally begin with the letters 'ev' (embedded vector).

The following table shows RTL conventions that are used in this chapter.

Table 20-1. RTL Notation

Notation	Meaning
x_{sf}	Signed fractional multiplication. Result of multiplying 2 quantities having bit lengths x and y taking the least significant x + y – 1 bits of the product and concatenating a 0 to the least significant bit forming a signed fractional result of x + y bits.
x_{si}	Signed integer multiplication
x_{su}, x_{sui}	Signed by Unsigned multiplication (same for int and frac)
x_{ui}	Unsigned integer multiplication
<<	Logical shift left. x << y shifts value x left by y bits, leaving zeros in the vacated bits.
>>	Logical shift right. x >> y shifts value x right by y bits, leaving zeros in the vacated bits.

20.3 SPE Programming Model

The z7260n3 core provides a register file with thirty-two 64-bit registers. The embedded category in the Power ISA instructions operate on the lower (least significant) 32 bits of the 64-bit register. SPE instructions generally take elements from each source register and operate on them with the corresponding elements of a second source register (and/or the accumulator) to produce results. Results are placed in the destination register and/or the accumulator. Vector instructions (i.e. produce results of more than one element)

provide results for each element that are independent of the computation of the other elements. These instructions can also be used to perform scalar DSP operations by ignoring the results of the upper 32-bit half of the register file.

SPE compare instructions and set instructions with record store the comparison result into the condition register (CR). The meaning of the CR bits are now overloaded for SPE operations. SPE compare instructions specify a CR field, two source registers, and the type of compare: greater than, less than, or equal. Two bits of the CR field are written with the result of the vector compare: one for each of the high and low 32-bits of the result. The remaining two bits reflect the ANDing and ORing of the vector compare results. An additional set of compare instructions (**evset$_{xx}$**[.]) return a set of Boolean values into a destination register, allowing for subsequent predicated computational operations, such as a select operation to be performed.

A partially visible accumulator register is architected for the SPE integer and fractional multiply accumulate forms of instructions. Its usage is described in Accumulator Register.

20.3.1 GPR Registers

The SPE requires a GPR register file with thirty-two 64-bit registers. For 32-bit implementations, the embedded category of the Power ISA instructions that normally operate on a 32-bit register, access and change only the least significant 32-bits of the GPRs. They leave the most significant 32-bits unchanged. SPE instructions view the 64-bit register as being composed of a vector of elements, each of which is 32 bits, 16 bits, or 8 bits wide.

Nomenclature is as follows:

- The most significant 32 bits are called word 0 (W0), the upper word, high word or even word.
- The least significant 32 bits are called word 1 (W1), the lower word, low word or odd word.
- Half word elements are called half word 0, 1, 2, or 3, from most significant to least significant.
- Byte elements are called byte 0, 1, 2, 3, 4, 5, 6, or 7, from most significant to least significant.

Unless otherwise specified, SPE instructions write all 64 bits of the destination register.

The following figure shows vector storage in GPRs.

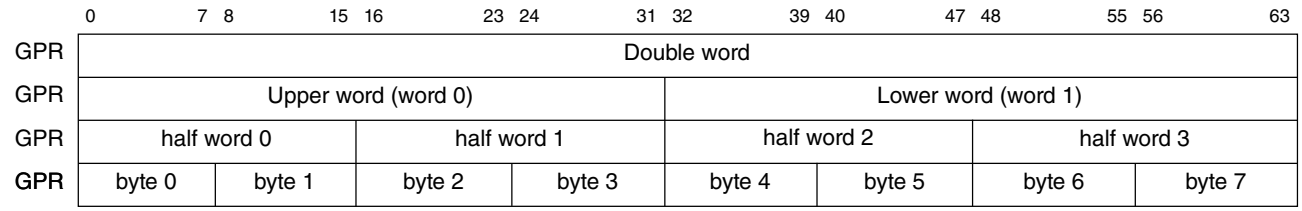


Figure 20-1. Vector Storage in GPRs

20.3.2 Accumulator Register

The accumulator is a 64-bit register that allows the back-to-back execution of dependent MAC and dot product instructions, something that is found in the inner loops of DSP code such as FIR and FFT filters. The accumulator is partially visible to the programmer in that its results do not have to be explicitly read to use them. Instead, they are always copied into a 64-bit destination GPR that is specified as part of the instruction. However, the accumulator has to be explicitly initialized when starting a new accumulation loop.

The accumulator is for used the following kinds of instructions:

- Certain integer/fractional accumulation
- Multiply accumulate (MAC)
- Dot product
- Summation forms

Based upon the type of instruction, the accumulator can hold either a single 64-bit value or a vector of two 32-bit elements, a vector of four 16-bit elements, or vector of eight 8-bit elements. In addition, for certain instructions, the accumulator can be updated along with the destination register.

The following figure shows accumulator storage.

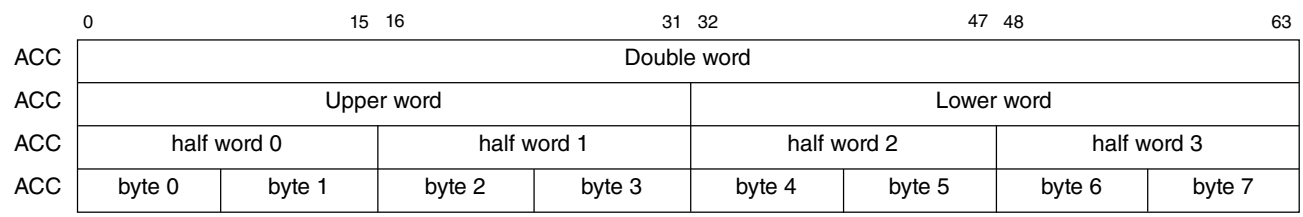


Figure 20-2. Accumulator Storage

An example of a MAC instruction is **evmhossfaawr**D**,r**A**,r**B. In this instruction, the least significant 16 bits of **r**A and **r**B are multiplied for both elements of the vector; the result is shifted left one bit and added to the accumulator; and the result is possibly saturated to 32 bits in case of overflow. The final result is placed both in the accumulator and also in **r**D. Therefore, the result of this instruction can be used by accessing **r**D.

To read the accumulator contents into a **register**, the **evmar** instruction is used. To initialize the accumulator, the **evmra** instruction or another instruction targeting the accumulator such as **evsplati$_{xx}$a** is used.

20.3.3 SPE Status and Control Register (SPEFSCR)

The z7260n3 core implements the SPEFSCR register for status reporting and control of SPE instructions. This register is also used by the embedded floating-point units. Status and control bits are shared for floating-point operations and SPE operations. The SPEFSCR register is implemented as special purpose register (SPR) number 512 and is read and written by the **mfspr** and **mtspr** instructions in both user and supervisor mode. SPE instructions affect both the high element (bits 0–1) and low element status flags (bits 16–17).

The following figure shows the SPEFSCR.

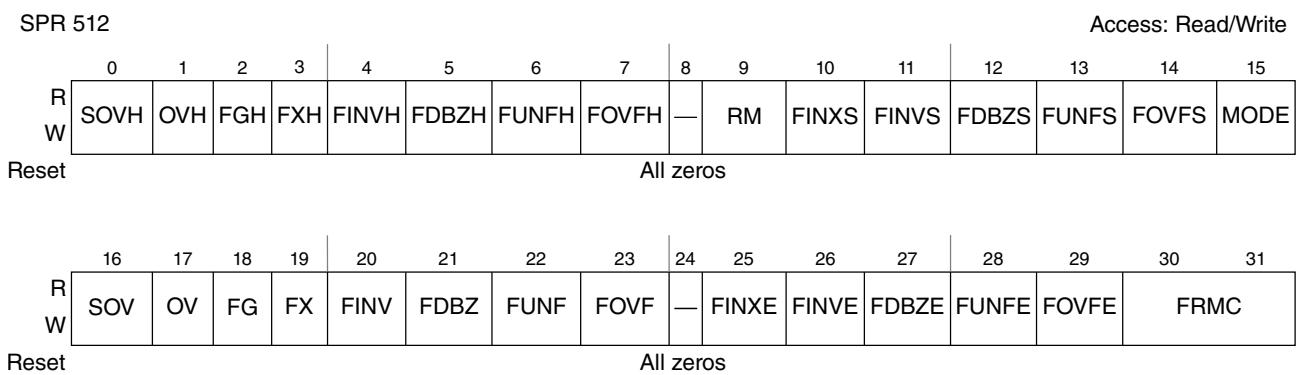


Figure 20-3. SPE/EFPU Status and Control Register (SPEFSCR)

The following table describes the SPEFSCR bits.

Table 20-2. SPE Status and Control Register

Bits	Name	Description
0 (32)	SOVH	Summary Integer Overflow High The SOVH bit is set to 1 whenever an instruction sets OVH. The SOVH bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
1 (33)	OVH	Integer Overflow High

Table continues on the next page...

Table 20-2. SPE Status and Control Register (continued)

Bits	Name	Description
		The OVH bit is set to 1 whenever an integer or fractional SPE instruction signals an overflow in the upper half of the result.
2 (34)	FGH	Embedded Floating-Point Guard bit High Defined by Embedded Floating-Point APUs.
3 (35)	FXH	Embedded Floating-Point Inexact bit High Defined by Embedded Floating-Point APUs.
4 (36)	FINVH	Embedded Floating-Point Invalid Operation/Input error High Defined by Embedded Floating-Point APUs.
5 (37)	FDBZH	Embedded Floating-Point Divide by Zero High Defined by Embedded Floating-Point APUs.
6 (38)	FUNFH	Embedded Floating-Point Underflow High Defined by Embedded Floating-Point APUs.
7 (39)	FOVFH	Embedded Floating-Point Overflow High Defined by Embedded Floating-Point APUs.
8 (40)	—	Reserved
9 (41)	RM	Rounding Mode - Fixed Point 0 Normal Rounding (Biased-rounding), rounding performed by adding 1/2 lsb 1 Round to Nearest Even Rounding (convergent rounding), round to nearest even value
10 (42)	FINXS	Embedded Floating-Point Inexact Sticky Flag Defined by Embedded Floating-Point APUs.
11 (43)	FINVS	Embedded Floating-Point Invalid Operation Sticky Flag Defined by Embedded Floating-Point APUs.
12 (44)	FDBZS	Embedded Floating-Point Divide by Zero Sticky Flag Defined by Embedded Floating-Point APUs.
13 (45)	FUNFS	Embedded Floating-Point Underflow Sticky Flag Defined by Embedded Floating-Point APUs.
14 (46)	FOVFS	Embedded Floating-Point Overflow Sticky Flag Defined by Embedded Floating-Point APUs.
15 (47)	MODE	Embedded Floating-Point Operating Mode Defined by Embedded Floating-Point APUs.
16 (48)	SOV	Summary Integer Overflow The SOV bit is set to 1 whenever an instruction sets OV. The SOV bit remains set until it is cleared by an **mtspr** instruction specifying the SPEFSCR register.
17 (49)	OV	Integer Overflow The OV bit is set to 1 whenever an integer or fractional SPE instruction signals an overflow in the low element result.
18 (50)	FG	Embedded Floating-Point Guard bit (low/scalar) Defined by Embedded Floating-Point APUs.

Table continues on the next page...

Table 20-2. SPE Status and Control Register (continued)

Bits	Name	Description
19 (51)	FX	Embedded Floating-Point Inexact bit (low/scalar) Defined by Embedded Floating-Point APUs.
20 (52)	FINV	Embedded Floating-Point Invalid Operation / Input error (low/scalar) Defined by Embedded Floating-Point APUs.
21 (53)	FDBZ	Embedded Floating-Point Divide by Zero (low/scalar) Defined by Embedded Floating-Point APUs.
22 (54)	FUNF	Embedded Floating-Point Underflow (low/scalar) Defined by Embedded Floating-Point APUs.
23 (55)	FOVF	Embedded Floating-Point Overflow (low/scalar) Defined by Embedded Floating-Point APUs.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-Point Round (Inexact) Exception Enable Defined by Embedded Floating-Point APUs.
26 (58)	FINVE	Embedded Floating-Point Invalid Operation / Input Error Exception Enable Defined by Embedded Floating-Point APUs.
27 (59)	FDBZE	Embedded Floating-Point Divide by Zero Exception Enable Defined by Embedded Floating-Point APUs.
28 (60)	FUNFE	Embedded Floating-Point Underflow Exception Enable Defined by Embedded Floating-Point APUs.
29 (61)	FOVFE	Embedded Floating-Point Overflow Exception Enable Defined by Embedded Floating-Point APUs.
30–31 (62–63)	FRMC	Embedded Floating-Point Rounding Mode Control Defined by Embedded Floating-Point APUs.

20.3.3.1 Context Switch

When a context switch occurs, the OS process must explicitly save the accumulator as part of the context of the swapped-out task and then explicitly load the accumulator from the context of the new task that is being swapped in. When the old task is restarted, its accumulator must be restored before restarting the task.

20.3.4 GPRs and Power ISA Instructions

The z7260n3 core implements the 32-bit forms of the embedded category instructions in the Power ISA. All 32-bit Power ISA instructions operate upon the lower half of the 64-bit GPR. These instructions do not affect the upper half of a GPR.

20.3.5 SPE Available Bit in MSR

MSR[SPE] is defined as the SPE available bit. If this bit is clear and software attempts to execute any of the SPE instructions other than the **brinc** instruction (which does not affect the upper 32-bits of a GPR), the SPE unavailable exception is taken. If this bit is set, software can execute any of the SPE instructions.

20.3.6 SPE Exception Bit in ESR

ESR[SPE] is defined as the SPE exception bit. This bit is set whenever the processor takes an exception related to the execution of the SPE instructions.

20.3.7 Data Formats

The SPE provides two different data formats, integer and fractional. Integer data formats can be treated as signed or unsigned quantities. Fractional data formats are usually treated as signed quantities

20.3.7.1 Integer Format

Integer data format is the same as what is conventionally used in computing.

Unsigned integers consist of 8, 16, 32, or 64-bit binary integer values. The largest representable value is $2^n – 1$ where n represents the number of bits in the value. The smallest representable value is 0. Certain computations that produce values larger than $2^n – 1$ or smaller than 0 set OV or OVH in the SPEFSCR.

Signed integers consist of 8, 16, 32, or 64-bit binary values in twos-complement form. The largest representable value is $2^{n-1} – 1$ where n represents the number of bits in the value. The smallest representable value is $–2^{n-1}$. Certain computations that produce values larger than $2^{n-1} – 1$ or smaller than $–2^{n-1}$ set OV or OVH in the SPEFSCR.

20.3.7.2 Fractional Format

Fractional data format is the same that is conventionally used for DSP fractional arithmetic. Fractional data is useful for representing data converted from analog devices.

Unsigned fractions consist of 16, 32, or 64-bit binary fractional values that range from 0 to less than 1. Unsigned fractions place the decimal point immediately to the left of the most significant bit. The most significant bit of the value represents the value 2^{-1}, the next most significant bit represents the value 2^{-2} and so on. The largest representable value is $1 – 2^{-n}$ where n represents the number of bits in the value. The smallest representable value is 0. Certain computations that produce values larger than $1 – 2^{-n}$ or smaller than 0 set OV or OVH in the SPEFSCR. SPE does not contain explicit instructions that manipulate unsigned fractional data. Unsigned integer forms produce the same bit exact results as unsigned fractional values would, therefore unsigned fractional instruction forms are not defined for SPE.

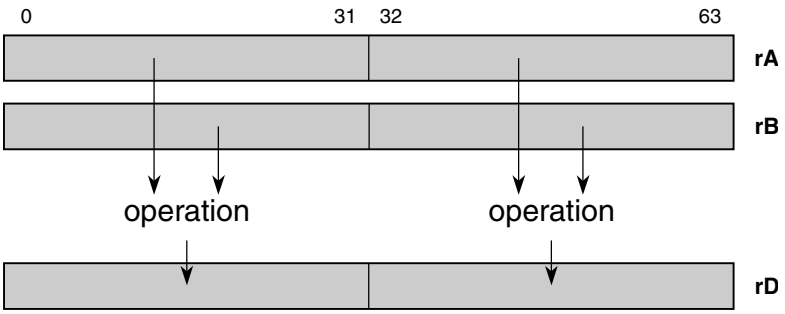
Signed fractions consist of 16, 32, or 64-bit binary fractional values in twos complement form that range from –1 to less than 1. Signed fractions in 1.31 or 1.63 format place the decimal point immediately to the right of the most significant bit. The largest representable value is $1 – 2^{-(n-1)}$ where n represents the number of bits in the value. The smallest representable value is –1. Certain computations that produce values larger than $1 – 2^{-(n-1)}$or smaller than –1 set OV or OVH in the SPEFSCR. Multiplication of two signed fractional values causes the result to be shifted left one bit to remove the resultant redundant sign bit in the product. In this case, a 0 bit is concatenated as the least significant bit of the shifted result.

Guarded fractional representations are also available in 33.31 format and in 17.47 format for a subset of operations, providing for significant guarding capabilities.

20.3.8 Computational Operations

SPE supports several different computational capabilities. Both modulo and saturation results can be performed. Modulo results produce truncation of the overflow bits in a calculation. Saturation provides a maximum or minimum representable value (for the data type) for overflow or underflow respectively. Instructions are provided for a wide range of computational capability. The operation types can be divided into several basic categories:

- Simple vector instructions. These instructions use the corresponding elements of the operands to produce a vector result that is placed in the destination register, the accumulator, or both.



- arithmetic, logical, shift, and rotate of vector elements
- Averaging, summation, rounding, min, max, sum of absolute differences, absolute differences, saturation, operations
- vector permutation, packing, unpacking, merge, swap, extraction, interleave, de-interleave operations

- Multiply and accumulate instructions. These instructions perform multiply operations, optionally add the result to the accumulator and place the result into the destination register and optionally into the accumulator. These instructions are composed of different multiply forms, data formats and data accumulate options.
- Dot product instructions. These instructions perform multiple multiply operations, optionally add the results to the accumulator, and place the result into the destination register and optionally into the Accumulator. These instructions are composed of different forms, data formats and data accumulate options.
- Load and store instructions. These instructions provide load and store capabilities for moving data to and from memory. A variety of forms are provided that position data for efficient computation.
- Compare instructions and set instructions.
- Miscellaneous instructions. These instructions perform miscellaneous functions such as field manipulation, bit-reversed and circular incrementing, count leading, and more.

20.3.8.1 Simple Vector Arithmetic Instructions

Simple vector arithmetic instructions are outlined in the following table.

Table 20-3. Simple Vector Arithmetic Instructions

Basic Operation	Variants	Description	ACC?
Absolute Value	evabsb, evabsh, evabs, evabsd	absolute value byte, half word, word, double word elements	—
	evabsbs, evabshs, evabss, evabsds	abs b, h, w, d with saturation	—
Absolute Difference	evabsdifsb, evabsdifsh, evabsdifsw, evabsdifub, evabsdifuh, evabsdifuw	absolute difference signed/unsigned byte, half word, word elements	—
Add	evaddb, evaddh, evaddw, evaddd	add byte, half word, word, double word elements	—
	evaddbss, evaddhss, evaddwss, evadddssevaddbus, evaddhus, evaddwus, evadddus	add byte, half word, word, double word elements with signed or unsigned saturation	—
	evaddhx, evaddhxss, evaddhxus	add exchanged half word elements with optional signed or unsigned saturation. The even and odd half word elements of operand rA are pairwise exchanged before adding	—
	evaddwx, evaddwxss, evaddwxus	add exchanged word elements with optional signed or unsigned saturation. The high and low word elements of operand rA are exchanged before adding	—
	evaddib, evaddih, evaddiw	add unsigned imm value UIMM to all elements	—
	evaddsmiaaw, evaddssiaaw, evaddumiaaw, evaddusiaaw	add word elements from rA and Accumulator using signed/unsigned modulo/saturation operations, results into rD and Accumulator	Y
	evaddsmiaa, evaddssiaa, evaddusiaa	add 64-bit value in rA and Accumulator with optional signed/unsigned saturation, result into rD and Accumulator	Y
AddSubf	evadd2subf2h, evadd2subf2hss	add for upper 2 half word elements, subf for lower 2 elements, with optional signed saturation.	—
	evaddsubfh, evaddsubfhss	add for even half word elements, subf for odd elements, with optional signed saturation.	—
	evaddsubfhx, evaddsubfhxss	The even and odd half word elements of operand rA are pairwise exchanged and then the resulting even elements are added and the odd elements are subtracted to/from elements in rB, with optional signed saturation.	—
	evaddsubfw, evaddsubfwss	The high word element of rA is added and the low word element of rA is subtracted to/from the corresponding element of rB, with optional signed saturation.	—
	evaddsubfwx, evaddsubfwxss	The word elements of rA are exchanged and then the resulting high word element is added and low word elements is subtracted to/from word elements of rB, with optional signed saturation.	—
Average	evavgbs, evavghs, evavgws, evavgds, evavgbsr, evavghsr, evavgwsr, evavgdsr evavgbu, evavghu, evavgwu, evavgdu evavgbur, evavghur, evavgwur, evavgdur	compute the average of corresponding elements in rA and rB, signed/unsigned with optional rounding	—

Table continues on the next page...

Table 20-3. Simple Vector Arithmetic Instructions (continued)

Basic Operation	Variants	Description	ACC?
Count Leading	evcntlsh, evcntlzh	count leading sign/zero bits in each half word	—
	evcntlsw, evcntlzw	count leading sign/zero bits in each word	
Divide	evdivws, evdivwu,	32 / 32 → 32 signed, unsigned integer	—
	evdivwsf, evdivwuf	32 / 32 → 32 signed, unsigned fractional	
	evdivs, evdivu	64 / 64 → 64, signed, unsigned	
Extend	evextsb, evextzb	the low byte of each word element in rA is sign/zero extended to a word and placed into rD	—
	evextsbh	the odd bytes of rA are sign extended to half words and placed into rD	—
	evextsh, *evextzh* (use evclrh)	the odd half word elements in rA are sign/zero extended to a word and placed into rD	—
	evextsw	the low word element in rA is sign extended to 64-bits and placed into rD	—
Maximum	evmaxbs, evmaxhs, evmaxws, evmaxds	maximum of elements in rA signed; b, h, w, d	—
	evmaxbu, evmaxhu, evmaxwu, evmaxdu	maximum of elements in rA unsigned; b, h, w, d	
	evmaxbpsh, evmaxbpuh	pairwise maximum of bytes in rA extended to half word, signed/unsigned	—
	evmaxhpsw, evmaxhpuw	pairwise maximum of half words in rA extended to word, signed/unsigned	—
	evmaxwpsd, evmaxwpud	pairwise maximum of words in rA extended to double word, signed/unsigned	—
Maximum Magnitude	evmaxmagws	pairwise maximum of magnitude values of signed words in rA	—
Minimum	evminbs, evminhs, evminws, evminds	minimum of elements in rA signed; b, h, w, d	—
	evminbu, evminhu, evminwu, evmindu	minimum of elements in rA unsigned; b, h, w, d	
	evminbpsh, evminbpuh	pairwise minimum of bytes in rA extended to half word, signed/unsigned	—
	evminhpsw, evminhpuw	pairwise minimum of half words in rA extended to word, signed/unsigned	—
	evminwpsd, evminwpud	pairwise minimum of words in rA extended to double word, signed/unsigned	—
Negate	evnegb, evnegh, evneg, evnegd	negate signed elements in rA; b,h,w,d	—
	evnegbs, evneghs, evnegs, evnegds	negate signed elements in rA with saturation; b,h,w,d	—
	evnegbo, evnegho, evnegwo	negate signed odd elements in rA; b,h,w	—
	evnegbos, evneghos, evnegwos	negate signed odd elements in rA with saturation; b,h,w	—
Round	evrndhb, evrndhbss, evrndhbus	The four half word elements of rA are rounded into 8-bits and placed into the even bytes of rD with optional signed or unsigned saturation	—

Table continues on the next page...

Table 20-3. Simple Vector Arithmetic Instructions (continued)

Basic Operation	Variants	Description	ACC?
	evrndhnb, evrndhnbss, evrndhnbus	The four half word elements of rA are rounded into 8-bits using round to nearest even rounding and placed into the even bytes of rD with optional signed or unsigned saturation	—
	evrndwh, evrndwhss, evrndwhus	The two word elements of rA are rounded into 16-bits and placed into the even half words of rD with optional signed or unsigned saturation	—
	evrndwnh, evrndwnhss, evrndwnhus	The two word elements of rA are rounded into 16-bits using round to nearest even rounding and placed into the even half words of rD with optional signed or unsigned saturation	—
	evrnddw, evrnddwss, evrnddwus	The double word element of rA is rounded into 32-bits and placed into the high word of rD with optional signed or unsigned saturation. The low word is cleared.	—
	evrndndw, evrndndwss, evrndndwus	The double word element of rA is rounded into 32-bits using round to nearest even rounding and placed into the high word of rD with optional signed or unsigned saturation. The low word is cleared.	—
Sum of Absolute Differences	evsad2sh, evsad2sha, evsad2shaaw	Sums of pairs of absolute differences of 2 signed half words, optionally loading the Accumulator, or accumulating with the Accumulator values	opt.
	evsad2uh, evsad2uha, evsad2uhaaw	Sums of pairs of absolute differences of 2 unsigned half words, optionally loading the Accumulator, or accumulating with the Accumulator values	opt.
	evsad4sb, evsad4sba, evsad4sbaaw	Sums of four absolute differences of 2 signed bytes, optionally loading the Accumulator, or accumulating with the Accumulator values	opt.
	evsad4ub, evsad4uba, evsad4ubaaw	Sums of four absolute differences of 2 unsigned bytes, optionally loading the Accumulator, or accumulating with the Accumulator values	opt.
	evsadsw, evsadswa, evsadswaa	Sum of pair of absolute differences of 2 signed words, optionally loading the Accumulator, or accumulating with the Accumulator value	opt.
	evsaduw, evsaduwa, evsaduwaa	Sum of pair of absolute differences of 2 unsigned words, optionally loading the Accumulator, or accumulating with the Accumulator value	opt.
Saturate	evsatsbub	Saturate signed byte to unsigned byte range	
	evsatubsb	Saturate unsigned byte to signed byte range	—
	evsatsdsw, evsatsduw	Saturate signed double word to signed or unsigned word range	—
	evsatuduw	Saturate unsigned double word to unsigned word range	—
	evsatshsb, evsatshub	Saturate signed half word to signed or unsigned byte range	—
	evsatshuh	Saturate signed half word to unsigned half word range	—
	evsatuhub	Saturate unsigned half word to unsigned byte range	—
	evsatuhsh	Saturate unsigned half word to signed half word range	—

Table continues on the next page...

Table 20-3. Simple Vector Arithmetic Instructions (continued)

Basic Operation	Variants	Description	ACC?
	evsatswgsdf	Saturate signed word guarded (17.47) to signed double word fractional (1.63) range	—
	evsatswsh, evsatswuh	Saturate signed word to signed or unsigned half word range	—
	evsatswuw	Saturate signed word to unsigned word range	—
	evsatuwuh	Saturate unsigned word to unsigned half word range	—
	evsatuwsw	Saturate unsigned word to signed word range	—
Subf	evsubfb, evsubfh, evsubfw, evsubfd	subtract byte, half word, word, double word elements	—
	evsubfbss, evsubfhss, evsubfwss, evsubfdss evsubfbus, evsubfhus, evsubfwus, evsubfdus	subtract byte, half word, word, double word elements with signed or unsigned saturation	—
	evsubfhx, evsubfhxss, evsubfhxus	subtract exchanged half word elements with optional signed or unsigned saturation. The even and odd half word elements of operand rA are pairwise exchanged before subtracting	—
	evsubfwx, evsubfwxss, evsubfwxus	subtract exchanged word elements with optional signed or unsigned saturation. The high and low word elements of operand rA are exchanged before subtracting	—
	evsubifb, evsubifh, evsubifw	subtract unsigned imm value UIMM from all elements	—
	evsubfsmiaaw, evsubfssiaaw, evsubfumiaaw, evsubfusiaaw	subtract word elements in rA from Accumulator using signed/unsigned modulo/saturation operations, results into rD and Accumulator	Y
	evsubfsmiaa, evsubfssiaa, evsubfusiaa	subtract 64-bit value in rA from Accumulator with optional signed/unsigned saturation, result into rD and Accumulator	Y
SubfAdd	evsubf2add2h, evsubf2add2hss	subtract for upper 2 half word elements, add for lower 2 elements, with optional signed saturation.	—
	evsubfaddh, evsubfaddhss	subtract for even half word elements, add for odd elements, with optional signed saturation.	—
	evsubfaddhx, evsubfaddhxss	The even and odd half word elements of operand rA are pairwise exchanged and then the resulting even elements are subtracted and the odd elements are added from/to elements in rB, with optional signed saturation.	—
	evsubfaddw, evsubfaddwss	The low word element of rA is added and the high word element of rA is subtracted to/from the corresponding element of rB, with optional signed saturation.	—
	evsubfaddwx, evsubfaddwxss	The word elements of rA are exchanged and then the resulting high word element is subtracted and low word element is added from/to word elements of rB, with optional signed saturation.	—
Summation/Diff	evsumws, evsumwu, evsumwsa, evsumwua	The signed or unsigned word elements of rA are summed together into 64 bits and placed into rD and optionally into the Accumulator	opt

Table continues on the next page...

Table 20-3. Simple Vector Arithmetic Instructions (continued)

Basic Operation	Variants	Description	ACC?
	evsumwsaa, evsumwuaa	The signed or unsigned word elements of rA are summed together along with the contents of the Accumulator and placed into rD and the Accumulator	Y
	evsum2hs, evsum2hu, evsum2hsa, evsum2hua	Signed or unsigned pairs of half word elements of rA are summed together into words and placed into rD and optionally into the Accumulator	opt
	evsum2hsaaw, evsum2huaaw	Signed or unsigned pairs of half word elements of rA are summed together along with the contents of the corresponding word element of the accumulator, into words, and placed into rD and the Accumulator	Y
	evsum4bs, evsum4bu, evsum4bsa, evsum4bua	Signed or unsigned quads of byte elements of rA are summed together into words and placed into rD and optionally into the Accumulator	opt
	evsum4bsaaw, evsum4buaaw	Signed or unsigned quads of byte elements of rA are summed together along with the contents of the corresponding word element of the accumulator, into words, and placed into rD and the Accumulator	Y
	evsum2his, evsum2hisa	Signed pairs of interleaved half word elements of rA are summed together into words and placed into rD and optionally into the Accumulator	opt
	evsum2hisaaw	Signed pairs of interleaved half word elements of rA are summed together along with the contents of the corresponding word element of the accumulator, into words, and placed into rD and the Accumulator	Y
	evdiff2his, evdiff2hisa	Signed pairs of interleaved half word elements of rA are subtracted to produce a pair of word differences and placed into rD and optionally into the Accumulator	opt
	evdiff2hisaaw	Signed pairs of interleaved half word elements of rA are subtracted to produce a pair of word differences and the differences are added together with the contents of the corresponding word element of the accumulator, into words, and placed into rD and the Accumulator	Y

20.3.8.1.1 Vector Logical Instructions

Vector logical instructions are outlined in the following table.

Table 20-4. Simple Vector Logical Instructions

Basic Operation	Variants	Description
AND	evand	AND word elements of rA and rB
ANDC	evandc	AND word elements of rA with complemented elements of rB

Table continues on the next page...

Table 20-4. Simple Vector Logical Instructions (continued)

Basic Operation	Variants	Description
Clear	evclrbe, evclrbo	Clear (zero) even bytes of source value in rA using immediate mask (mask). Clear (zero) odd bytes of source value in rA using immediate mask (mask).
	evclrh	Clear (zero) half word elements of source value in rA using immediate mask (mask).
NAND	evnand	NAND word elements of rA and rB
NOR	evnor	NOR word elements of rA and rB
OR	evor	OR word elements of rA and rB
ORC	evorc	OR word elements of rA with complemented elements of rB
XNOR	eveqv	XNOR word elements of rA and rB
XOR	evxor	XOR word elements of rA and rB

20.3.8.1.2 Vector Shift/Rotate Instructions

Vector shift and rotate instructions are outlined in the following table.

Table 20-5. Simple Vector Shift/Rotate Instructions

Basic Operation	Variants	Description
Shift Left	evslb, evslh, evslw, evslev slbi, evslhi, evslwi, evsli	Logical shift left of the 8,16, 32 or 64-bit element(s) in rA by the amount(s) in rB or by the immediate value UIMM
	evsloi	Logical shift left of the value in rA by 0 to 7 bytes
Logical Shift Right	evsrbu, evsrhu, evsrwu, evsru evsrbiu, evsrhiu, evsrwiu, evsriu	Logical shift right of the 8,16, 32 or 64-bit element(s) in rA by the amount(s) in rB or by the immediate value UIMM
	evsroiu	Logical shift right of the value in rA by 0 to 7 bytes
Arithmetic Shift Right	evsrbs, evsrhs, evsrws, evsrs evsrbis, evsrhis, evsrwis, evsris	Arithmetic shift right of the 8,16, 32 or 64-bit element(s) in rA by the amount(s) in rB or by the immediate value UIMM
	evsrois	Arithmetic shift right of the value in rA by 0 to 7 bytes
Rotate Left	evrlb, evrlh, evrlw evrlbi, evrlhi, evrlwi	Rotate left of the 8,16, or 32-bit elements in rA by the amount(s) in rB or by the immediate value UIMM

20.3.8.1.3 Vector Compare and Vector Set Instructions

Vector compare and set instructions are outlined in Table 20-6 and Table 20-7. The compare operations update the condition register with the results of the comparison.

Table 20-6. Vector Compare Instructions

Basic Comparison Operation	Variants	Description
=	evcmpeq, evcmpeqd	Compare word or double word elements for equal
>	evcmpgts, evcmpgtu, evcmpgtds, evcmpgtdu	Compare word or double word elements for greater than signed/ unsigned
<	evcmplts, evcmpltu, evcmpltds, evcmpltdu	Compare word or double word elements for less than signed/ unsigned

Table 20-7. Vector Set Instructions

Basic Comparison Operation	Variants	Description
=	evseteqb[.], evseteqh[.], evseteqw[.]	Compare byte, half word or word elements in rA and rB for equal. For each byte, half word or word, set destination byte half word or word to all '1's if condition met. Optionally set CR0 with comparison results.
>	evsetgtbs[.], evsetgtbu[.], evsetgths[.], evsetgthu[.], evsetgtws[.], evsetgtwu[.]	Compare byte, half word or word elements in rA and rB for greater than signed or unsigned. For each byte, half word or word, set destination byte half word or word to all '1's if condition met. Optionally set CR0 with comparison results.
<	evsetltbs[.], evsetltbu[.], evsetlths[.], evsetlthu[.], evsetltws[.], evsetltwu[.]	Compare byte, half word or word elements in rA and rB for greater than signed or unsigned. For each byte, half word or word, set destination byte half word or word to all '1's if condition met. Optionally set CR0 with comparison results.

20.3.8.1.4 Vector Select Instructions

Vector select instructions are outlined in the following table.

Table 20-8. Vector Select Instructions

Operation	Variants	Description
Select	evsel	Select word elements from rA or rB based on crS condition register field
Select Bits	evselbit	Select bit elements from rA or rB based on select bit vector in rD, place results into rD
	evselbitm0	Insert bit elements from rB into rD based on select bit mask in rA of 0, place results into rD
	evselbitm1	Insert bit elements from rB into rD based on select bit mask in rA of 1, place results into rD

20.3.8.1.5 Vector Data Arrangement Instructions

Vector data arrangement instructions are outlined in the following table. These instructions are used to rearrange fields of elements from one or more source vector registers.

Table 20-9. Vector Data Arrangement Instructions

Basic Operation	Variants	Description
De-interleave	evdlveb	de-interleave even bytes; the vector of even byte elements in rA and even byte elements in rB are concatenated and placed into rD
	evdlveob	de-interleave even/odd bytes; the vector of even byte elements in rA and odd byte elements in rB are concatenated and placed into rD
	evdlvob	de-interleave odd bytes; the vector of odd byte elements in rA and odd byte elements in rB are concatenated and placed into rD
	evdlvoeb	de-interleave odd/even bytes; the vector of odd byte elements in rA and even byte elements in rB are concatenated and placed into rD
	evdlveh	de-interleave even half words; the even half word elements in rA and even half word elements in rB are concatenated and placed into rD
	evdlveoh	de-interleave even/odd half words; the even half word elements in rA and odd half word elements in rB are concatenated and placed into rD
	evdlvoh	de-interleave odd half word; the odd half word elements in rA and odd half word elements in rB are concatenated and placed into rD
	evdlvoeh	de-interleave odd/even half words; the odd half word elements in rA and even half word elements in rB are concatenated and placed into rD
Interleave	evilveh	interleave even half words; the even half words from rA are placed into the even half words of rD and the even half words of rB are placed into the odd half words of rD
	evilveoh	interleave even/odd half words; the even half words from rA are placed into the even half words of rD and the odd half words of rB are placed into the odd half words of rD
	evilvhih	interleave high half words; the high half words from rA are placed into the even half words of rD and the high half words of rB are placed into the odd half words of rD
	evilvhiloh	interleave high/low half words; the high half words from rA are placed into the even half words of rD and the low half words of rB are placed into the odd half words of rD
	evilvloh	interleave low half words; the low half words from rA are placed into the even half words of rD and the low half words of rB are placed into the odd half words of rD
	evilvlohih	interleave low/high half words; the low half words from rA are placed into the even half words of rD and the high half words of rB are placed into the odd half words of rD
	evilvoeh	interleave odd/even half words; the odd half words from rA are placed into the even half words of rD and the even half words of rB are placed into the odd half words of rD

Table continues on the next page...

Table 20-9. Vector Data Arrangement Instructions (continued)

Basic Operation	Variants	Description
	evilvoh	interleave odd half words; the odd half words from rA are placed into the even half words of rD and the odd half words of rB are placed into the odd half words of rD
Merge	evmergehi	merge high words; the high word from rA is placed into the high word of rD and the high word of rB is placed into the low word of rD
	evmergehilo	merge high/low words; the high word from rA is placed into the high word of rD and the low word of rB is placed into the low word of rD
	evmergelo	merge low words; the low word from rA is placed into the high word of rD and the low word of rB is placed into the low word of rD
	evmergelohi	merge low/high words; the low word from rA is placed into the high word of rD and the high word of rB is placed into the low word of rD
Permute	evperm	Permute the byte elements in rB according to the permute vector in rA and place the results in rD
	evperm2	Permute the vector of concatenated byte elements from rA and rB according to the permute vector in rD and place the results in rD
	evperm3	Permute the vector of concatenated byte elements from rD and rB according to the permute vector in rA and place the results in rD
Pack	evpksdsws, evpkuduws	Pack the signed or unsigned double word elements from rA and rB into a pair of signed or unsigned word elements in rD, saturating if necessary
	evpksdswfrs	Pack the signed double word fractional elements from rA and rB into a pair of signed word elements in rD using the current rounding mode in SPEFSCR, saturating if necessary
	evpksdshefrs	Pack the signed 33.31 guarded fractional elements from rA and rB into a pair of signed half word even elements in rD using the current rounding mode in SPEFSCR, saturating if necessary
	evpkshsbs, evpkshubs, evpkuhubs	Pack the 8 signed or unsigned half word elements from rA and rB into 8 signed or unsigned byte elements in rD, saturating if necessary
	evpkswgshefrs	Pack the signed 17.47 guarded fractional elements from rA and rB into a pair of signed half word even elements in rD using the current rounding mode in SPEFSCR, saturating if necessary
	evpkswgswfrs	Pack the signed 17.47 guarded fractional elements from rA and rB into a pair of signed word elements in rD using the current rounding mode in SPEFSCR, saturating if necessary
	evpkswshs, evpkswuhs, evpkuwuhs	Pack the 4 signed or unsigned word elements from rA and rB into 4 signed or unsigned half word elements in rD, saturating if necessary
	evpkswshilvs	Pack the 4 signed word elements from rA and rB into 4 signed half word elements in rD with interleaving, saturating if necessary
	evpkswshfrs	Pack the 4 signed fractional word elements from rA and rB into 4 signed or fractional half word elements in rD using the current rounding mode in SPEFSCR, saturating if necessary
	evpkswshilvfrs	Pack the 4 signed fractional word elements from rA and rB into 4 signed or fractional half word elements in rD with interleaving using the current rounding mode in SPEFSCR, saturating if necessary
Splat	evsplatb	splat (replicate) the byte from rA selected by the immediate field into all byte elements of rD

Table continues on the next page...

Table 20-9. Vector Data Arrangement Instructions (continued)

Basic Operation	Variants	Description
	evsplath	splat (replicate) the half word from rA selected by the immediate field into all half word elements of rD
	evsplatfib, splatfih, splatfi	Splat the 5-bit SIMM field as a signed fraction into all byte, half word, or word elements of rD
	evsplatfiba, splatfiha, splatfia	Splat the 5-bit SIMM field as a signed fraction into all byte, half word, or word elements of rD and the accumulator
	evsplatfid	Splat the 5-bit SIMM field as a signed fraction into rD
	evsplatfida	Splat the 5-bit SIMM field as a signed fraction into rD and the accumulator
	evsplatfibo, splatfiho, splatfio	Splat the 5-bit SIMM field as a signed fraction into the odd byte, half word, or word elements of rD
	evsplatfiboa, splatfihoa, splatfioa	Splat the 5-bit SIMM field as a signed fraction into the odd byte, half word, or word elements of rD and the accumulator
	evsplatib, evsplatih, evsplati	Splat the 5-bit SIMM field as a signed integer into all byte, half word, or word elements of rD
	evsplatiba, evsplatiha, evsplatia	Splat the 5-bit SIMM field as a signed integer into all byte, half word, or word elements of rD and the accumulator
	evsplatid	Splat the 5-bit SIMM field as a signed integer into rD
	evsplatida	Splat the 5-bit SIMM field as a signed integer into rD and the accumulator
	evsplatibe, evsplatihe, evsplatie	Splat the 5-bit SIMM field as a signed integer into the even byte, half word, or word elements of rD
	evsplatibea, evsplatihea, evsplatiea	Splat the 5-bit SIMM field as a signed integer into the even byte, half word, or word elements of rD and the accumulator
Swap	evswapbhilo	bytes within the upper 2 byte pairs in rA are swapped, and concatenated with swapped bytes in the lower 2 byte pairs of rB.
	evswapblohi	bytes within the lower 2 byte pairs in rA are swapped, and concatenated with swapped bytes in the upper 2 byte pairs of rB.
	evswaphe	The even half words in rA are swapped, and merged with the odd half words of rB.
	evswaphhi	The upper 2 half words in rA are swapped, and concatenated with the lower 2 half words of rB.
	evswaphhilo	The upper 2 half words in rA are swapped, and concatenated with swapped lower 2 half words of rB.
	evswaphlo	The lower 2 half words in rA are swapped, and concatenated after the upper 2 half words of rB.
	evswaphlohi	The lower 2 half words in rA are swapped, and concatenated with swapped upper 2 half words of rB.
	evswapho	The odd half words in rA are swapped, and then merged with even half words of rB.
Unpack	evunpkhibsi, evunpkhibui, evunpklobsi, evunpklobui	Unpack the high or low 4 bytes of rA into signed or unsigned integer half words
	evunpkhihf, evunpkhihsi, evunkpkhihui, evunpklohf, evunpklohsi, evunkpklohui	Unpack the high or low 2 half words of rA into signed fractional, signed integer, or unsigned integer words

Table continues on the next page...

Table 20-9. Vector Data Arrangement Instructions (continued)

Basic Operation	Variants	Description
	evunpkhiwgsf, evunpklowgsf	Unpack the high or low word of rA into guarded signed fractional (17.47) format
Extract	evxtrb	A specified byte in rA is placed into a specified byte of rD, zeroing all other bytes of rD
	evxtrd	a double word is extracted from the concatenated byte elements of rA and rB and placed into rD
	evxtrh	A specified half word in rA is placed into a specified half word of rD, zeroing all other half words of rD
Insert	evinsb	A specified byte in rA is placed into a specified byte of rD; all other bytes of rD are unchanged.
	evinsh	A specified half word in rA is placed into a specified half word of rD; all other half words of rD are unchanged.

20.3.8.1.6 Multiply and accumulate instructions

These instructions perform multiply operations, optionally add the result to the accumulator and place the result into the destination register and optionally into the accumulator. These instructions are composed of different multiply forms, data formats and data accumulate options. The mnemonics for these instructions indicate their various characteristics. These are shown in the following table.

Table 20-10. Mnemonic Extensions for Multiply Accumulate Instructions

Extension	Meaning	Comments
Multiply Form		
he	half word even	16 × 16 → 32
heg	half word even guarded	16 × 16 → 32, 64-bit final accumulate result
ho	half word odd	16 × 16 → 32
hog	half word odd guarded	16 × 16 → 32, 64-bit final accumulate result
w	word	32 × 32 → 64
wehg	word even high guarded	32 × 32 → 64 in 17.47 format
wh	word high	32 × 32 → 32 (high order 32 bits of product)
wl	word low	32 × 32 → 32 (low order 32 bits of product)
wohg	word odd high guarded	32 × 32 ® 64 in 17.47 format
Data Format		
smf	signed modulo fractional	modulo, no saturation or overflow
smfr	signed modulo fractional round	modulo, no saturation or overflow, rounding based on current rounding mode
smi	signed modulo integer	modulo, no saturation or overflow
ssf	signed saturate fractional	saturation on product and accumulate

Table continues on the next page...

Table 20-10. Mnemonic Extensions for Multiply Accumulate Instructions (continued)

Extension	Meaning	Comments
ssfr	signed saturate fractional round	saturation on product and accumulate, rounding based on current rounding mode
ssi	signed saturate integer	saturation on accumulate
umi	unsigned modulo integer	modulo, no saturation or overflow
usi	unsigned saturate integer	saturation on accumulate
Accumulate Option		
a	place in Accumulator	result → rD, Accumulator
aa	add to Accumulator	Accumulator + result → rD, Accumulator
aaw	add to Accumulator as word elements	Accumulator[0:31] + result[0:31] → rD[0:31], Accumulator[0:31] Accumulator[32:63] + result[32:63] → rD[32:63], Accumulator[32:63]
aaw3	add to rD as word elements	rD[0:31] + result[0:31] → rD[0:31], Accumulator[0:31] rD[32:63] + result[32:63] → rD[32:63], Accumulator[32:63]
an	add negated to Accumulator	Accumulator – result → rD, Accumulator
anw	add negated to Accumulator as word elements	Accumulator[0:31] – result[0:31] → rD[0:31], Accumulator[0:31] Accumulator[32:63] – result[32:63] → rD[32:63], Accumulator[32:63]
anw3	add negated to rD as word elements	rD[0:31] – result[0:31] → rD[0:31], Accumulator[0:31] rD[32:63] – result[32:63] → rD[32:63], Accumulator[32:63]

20.3.8.1.7 Dot product instructions

These instructions perform multiple multiply operations, optionally add the results to the accumulator, and place the result into the destination register and optionally into the accumulator. These instructions are composed of different forms, data formats and data accumulate options. The mnemonics for these instructions indicate their various characteristics. These are shown in the following table.

Table 20-11. Mnemonic Extensions for Dot Product Instructions

Extension	Meaning	Comments
Multiply Form		
b	byte	8 × 8 + 8 × 8 + 8 × 8 + 8 × 8 → 32, high and low
4h	four half words	16 × 16 + 16 × 16 + 16 × 16 + 16 × 16 → 32
4hg	four half words guarded	16 × 16 + 16 × 16 + 16 × 16 + 16 × 16 → 64
h	half word	16 × 16 op 16 × 16 → 32, high and low
hih	high half words	16 × 16 op 16 × 16 → 32, high half words, used for complex mul
loh	low half words	16 × 16 op 16 × 16 → 32, low half words, used for complex mul

Table continues on the next page...

Table 20-11. Mnemonic Extensions for Dot Product Instructions (continued)

Extension	Meaning	Comments
4hxga	four half words exchanged guarded add	(16 × 16 + 16 × 16) + (16 × 16 + 16 × 16) → 64, even and odd rA half words changed
4hxgs	four half words exchanged guarded subtract	(16 × 16 - 16 × 16) + (16 × 16 - 16 × 16) → 64, even and odd rA half words changed
w	word	32 × 32 op 32 × 32 → 64
wg	word guarded	32 × 32 op 32 × 32 → 64 in 17.47 fractional format
wxga	word exchanged guarded add	32 × 32 + 32 × 32 → 64 in 17.47 fractional format, words in rA are changed
wxgs	word exchanged guarded subtract	32 × 32 - 32 × 32 → 64 in 17.47 fractional format, words in rA are changed
Operation		
a	add	addition of intermediate products
s	subtract	subtraction of intermediate products
c	complex	complex format arithmetic
Data Format		
smf	add signed modulo fractional	modulo, no saturation or overflow
smi	signed modulo integer	modulo, no saturation or overflow
ssf	signed saturate fractional	saturation on product and accumulate
ssfr	signed saturate fractional round	saturation on product and accumulate, rounding based on current rounding mode
ssi	signed saturate integer	saturation on product and accumulate
umi	unsigned modulo integer	modulo, no saturation or overflow
usi	unsigned saturate integer	saturation on product and accumulate
Accumulate Option		
a	place in Accumulator	result → rD, Accumulator
aa	add to Accumulator	Accumulator + result → rD, Accumulator
aa3	add to Accumulator, 3op	rD + result → rD, Accumulator
aaw	add to Accumulator as word elements	Accumulator[0:31] + result[0:31] → rD[0:31], Accumulator[0:31] Accumulator[32:63] + result[32:63] → rD[32:63], Accumulator[32:63]
aaw3	add to Accumulator as word elements, 3 op	rD[0:31] + result[0:31] → rD[0:31], Accumulator[0:31] rD[32:63] + result[32:63] → rD[32:63], Accumulator[32:63]

20.3.8.1.8 Miscellaneous Vector Instructions

Miscellaneous vector instructions are outlined in Table 20-4.

Table 20-12. Misc. Vector Instructions

Operation	Variants	Description
load vector for shift	evlvsl	load vector for shift left; place a vector of constant values for a vector permute for left shift
	evlvsr	load vector for shift right; place a vector of constant values for a vector permute for right shift
store Accumulator	evmar	move Accumulator to register rA
load Accumulator	evmra	move register rA to Accumulator
Bit reversed increment	brinc	Compute a bit-reversed increment for a memory offset for bit-reversed addressing
Circular Increment	circinc	Computes a modulo increment for supporting circular buffer index pointer modification

20.3.8.2 Load and Store Instructions

SPE provides a number of load and store instructions. These instructions provide load and store capabilities for moving data elements between the GPRs and memory. Data elements of 8, 16, 32, and 64 bits are supported. A variety of forms are provided that position data for efficient computation.

20.3.8.2.1 Addressing Modes—Non-Update forms

Base + index and base + scaled immediate addressing modes are provided. Base registers hold 64-bit pointer values (32-bit pointers in a 32-bit implementation of the architecture), while registers used as index values provide 32-bit index values. Scaled immediate values are unsigned and are scaled by the size of the access.

20.3.8.2.1.1 Base + Scaled Immediate Addressing—Non-Update Form

In the base + scaled immediate addressing mode, register rA holds a 32-bit pointer value or a value of zero (if rA = 0), and an immediate field in the instruction word provides a 5-bit unsigned immediate value which is zero-extended and scaled (shifted left) by 1, 2, or 3, depending on the size (half word, word, or double word) of the access. The sum of the value in rA and the zero-extended scaled immediate form the effective address:

```
if (rA = 0) then b ← 0
else b ← (rA32:63)
SCL ← {1,2,3} // half word, word, or double word
EA ← b + EXTZ(UIMM*SCL)
```

20.3.8.2.1.2 Base + Index Addressing

In the Base + Index addressing mode, register rA holds a 32-bit pointer value or a value of zero (if rA = 0), while register rB provides a 32-bit index. The sum forms the effective address:

```
if (rA = 0) then b ← 0
else b ← (rA32:63)
EA ← b + (rB)
```

20.3.8.2.2 Addressing Modes—Update forms

The base + scaled immediate addressing mode is also provided with an update form. As in the non-update form, base register rA holds 32-bit pointer values. For the update form of the base + scaled immediate addressing mode, the same effective address calculation is used as defined in Base + Scaled Immediate Addressing—Non-Update Form, and the calculated effective address is placed into rA by the instruction.

For the base + scaled immediate with update addressing mode, scaled immediate values of 0 are reserved for future definition and are treated as illegal. Instruction encodings with rA = 0 are also reserved for future definition and treated as illegal instructions.

20.3.8.2.3 Addressing Modes—Modify forms

The base + index addressing mode is also provided with a set of modify forms. In the modify forms, register rB holds 32-bit pointer values, while register rA is used to provide an index value as well as to provide specialized control information for performing a post-modification to the lower 32 bits of rA.

Modify forms are provided to allow for parallel address computations to occur, which are useful for sequential accessing of arrays, lists, circular buffers, and other complex data structures. Modify forms of load and store instructions cause a calculated update value to be placed in the lower portion of register rA. Support for specialized addressing modes are available when using base + index modify forms.

For the base + index modify forms, the modify calculation mode selection is based on a **mode** field in register rA (rA[0:3]). Modify forms modify the original value in rA based on an addressing calculation performed in parallel with the load or store instruction, which may or may not be the value of the effective address of the load or store instruction, depending on the actual calculation mode. This is in contrast to normal update forms of the Power Arch load and store instructions since the new value placed into rA need not correspond to the effective address of the load or store.

The following three modify calculation modes are currently defined and selected by the value in rA[0:3]:

- Linear addressing: mode = 0000
- Circular addressing: mode = 1000
- Bit-reversed addressing: mode = 1010

All other mode encodings are reserved, and either result in an unimplemented instruction exception, or a boundedly undefined result depending on the implementation.

Instruction encodings with rA = 0 are reserved for future definition and are treated as illegal instructions.

20.3.8.2.3.1 Linear Addressing Update Mode

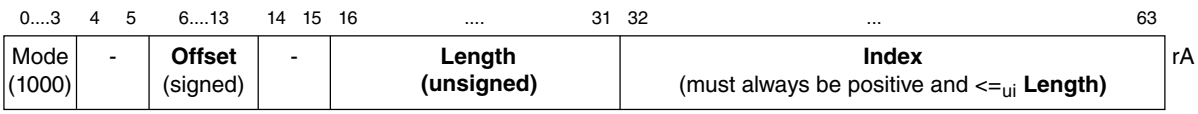
Linear addressing update calculation mode causes the sum of rA[32:63] and rB[32:63] to be placed into rA[32:63]:

```
if(mode=0000) then
   rA32:63 ← rA32:63 + rB32:63
```

20.3.8.2.3.2 Circular Addressing Modify Mode

Circular addressing modify mode is provided to support addressing of circular buffers. Circular addressing mode causes a circular increment to be performed on a portion of rA[32:63] (the circular buffer index portion of rA) after the EA calculation, using the offset and length specifiers in rA and the result is placed into rA[32–63]. rA[0–31] is left unchanged. rA[32–63] must be $\geq_{si}$ 0 and $\leq_{ui}$ Length, and the magnitude of Offset must be ≤ Length + 1, or the resulting value is boundedly undefined. rB must point to a double-word boundary in memory, and Length + 1 must be a multiple of eight bytes or an alignment error will be generated.

The following figureshows how rA is used in forming the update value for mode 1000 (circinc).



Listing 20-1. rA Used to Form Update Value for Mode 1000

```
Offset0:7 ← rA8:15;  //  signed byte offset, must be <= Length+1
Length0:15 ← rA16:31;  // unsigned buffer length-1 in bytes. Length is byte index of
                       // last byte in buffer.
                       // buffer must be aligned on a doubleword boundary, and be a
                       // multiple of 8 bytes, i.e. Length13:15 =3'b111.
Index0:31 ← rA32:63; // index into buffer, must be <=ui Length0:15, (so always >=si 0).

if ((Offset0 = 0) & ((EXTS32(Offset0:7) + Index0:31) >ui EXTZ32(Length0:15))) then
    rA32:63 ← Index0:31 + EXTS32(Offset0:7) - EXTZ32(Length0:15) - 1; // wrap at end

elseif (Offset0 = 1) & ((EXTS32(Offset0:7) + Index0:31) <si 0)) then
    rA32:63 ← Index0:31 + EXTS32(Offset0:7) + EXTZ32(Length0:15) + 1; // wrap at start
```

```
else rA32:63 ← Index0:31 + EXTS32(Offset0:7);
```

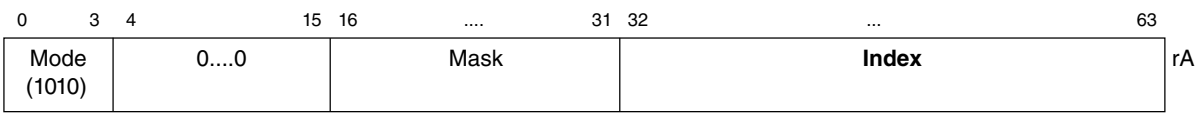
Note that **misalignment** may cause the operand fetched to span the virtual boundary between the last byte of the buffer at byte Buffer[Length] and the first byte of the Buffer at byte Buffer[0].

20.3.8.2.3.3 Bit-Reversed Addressing Modify Mode

Bit-reversed addressing modify calculation mode is provided to support addressing of buffers and arrays used in FFT calculations.

When using bit-reversed addressing modify mode, a bit-reversed increment is performed on rA[32:63] after the EA calculation, using a mask specifier in rA. The mask specifier is also used to indicate the bits of rA[32:63] which are updated.

Listing 20-2 on page 875 shows how rA is used in forming the update value for bit-reversed addressing update mode. Note that the computation is similar to the **brinc** instruction computation, but the mask is applied to updating only those bits of rA indicated by a '1' in the mask value, unlike in the **brinc** instruction, in which all low order bits of rD corresponding to the maximum mask size are updated.



Listing 20-2. rA Used to Form Update Value for Bit-Reversed Addressing Update Mode

```
Mask ← rA16:31
// mask value is log2(#points)1, zero extended, then left shifted log2(element size
// in bytes). e.g., a 16 point FFT on half words has a mask of 16'b0000000000011110

a ← rA48:63 // up to 64KB in a single FFT
d ← bitreverse(1 + bitreverse(a | ~Mask)))
rA32:63 ← rA32:47 || ((rA48:63 & ~Mask) |(d & Mask)) // different than brinc. allows main
pointer sharing to multiple buffers less than 64KB in size.
```

20.3.8.2.4 Vector Load and Store Instruction Summary

Vector load and store instructions are provided to load and store various size vectors of byte, half-word, word or double-word size. These instructions allow for endian-neutral code to be written. In addition, update forms of the non-indexed instructions are provided to allow for base register updates. Variations of the load instructions provide splat (replication) capability for placing a smaller vector element into multiple element positions in a vector register.

Vector load and store instructions are outlined in the following table.

Table 20-13. Vector Load and Store Instructions

Operation	Variants	Description
Load Byte	evlbbsplatb, evlbbsplatbu, evlbbsplatbx, evlbbsplatbmx	load byte and splat byte into 8 byte element positions
Load Double Word	evldb, evldbu, evldbx, evldbmx	load double word as byte elements
	evldd, evlddu, evlddx, evlddmx	load double word as double word
	evldh, evldhu, evldhx, evldhmx	load double word as half word elements
	evldw, evldwu, evldwx, evldwmx	load double word as word elements
Load Half Word	evlhhesplat, evlhhesplatu, evlhhesplatx, evlhhesplatmx	load half word into even half word elements, zeroing the odd half word s elements
	evlhhossplat, evlhhossplatu, evlhhossplatx, evlhhossplatmx	load half word into odd half word elements, sign-extending to word elements
	evlhhousplat, evlhhousplatu, evlhhousplatx, evlhhousplatmx	load half word into odd half word elements, zero-extending to word elements
	evlhhsplath, evlhhsplathu, evlhhsplathx, evlhhsplathmx	load half word into all half word elements
Load Word	evlwbe, evlwbeu, evlwbex, evlwbemx	load word as four byte elements into the four even byte elements, zeroing the odd byte elements
	evlwbos, evlwbosu, evlwbosx, evlwbosmx	load word as four byte elements into the four odd byte elements, sign-extending to half word elements
	evlwbou, evlwbouu, evlwboux, evlwboumx	load word as four byte elements into the four odd byte elements, zero-extending to half word elements
	evlwbsplatw, evlwbsplatwu, evlwbsplatwx, evlwbsplatwmx	load word as four byte elements into both word elements
	evlwhe, evlwheu, evlwhex, evlwhemx	load word as two half word elements into the two even half word elements, zeroing the odd half word elements
	evlwhos, evlwhosu, evlwhosx, evlwhosmx	load word as two half word elements into the two odd half word elements, sign-extending to word elements
	evlwhou, evlwhouu, evlwhoux, evlwhoumx	load word into the two odd half word elements, zero-extending to word elements
	evlwhsplat, evlwhsplatu, evlwhsplatx, evlwhsplatmx	load word as two half word elements, placing the first half word into both upper half word elements, second half word into both lower half word elements
	evlwhsplatw, evlwhsplatwu, evlwhsplatwx, evlwhsplatwmx	load word as two half word elements, into both word elements
	evlwwsplat, evlwhsplatu, evlwhsplatx, evlwhsplatmx	load word as word element, into both word elements
Store Double Word	evstdb, evstdbu, evstdbx, evstdbmx	store double word as byte elements
	evstdd, evstddu, evstddx, evstddmx	store double word as double word
	evstdh, evstdhu, evstdhx, evstdhmx	store double word as half word elements
	evstdw, evstdwu, evstdwx, evstdwmx	store double word as word elements
Store Half Word	evsthb, evsthbu, evsthbx, evsthbmx	store half word as byte elements
Store Word	evstwb, evstwbu, evstwbx, evstwbmx	store word as four byte elements
	evstwbe, evstwbeu, evstwbex, evstwbemx	store word from four even byte elements
	evstwbo, evstwbou, evstwbox, evstwbomx	store word from four odd byte elements
	evstwhe, evstwheu, evstwhex, evstwhemx	store word from two even half word elements

Table continues on the next page...

Table 20-13. Vector Load and Store Instructions (continued)

Operation	Variants	Description
	evstwho, evstwhou, evstwhox, evstwhomx	store word from two odd half word elements
	evstwwe, evstwweu, evstwwex, evstwwemx	store word from even word element
	evstwwo, evstwwou, evstwwox, evstwwomx	store word from odd word element

20.3.8.3 SPE Exceptions

The architecture defines the following SPE exceptions:

- SPE unavailable exception
- SPE vector alignment exception

Interrupt vector offset registers (IVOR) IVOR32 (SPE/embedded floating point unavailable interrupt) and IVOR5 (alignment interrupt), are used by the interrupt model. The SPR number for IVOR32 is 528, IVOR5 is defined by Power ISA. These registers are privileged.

20.3.8.3.1 SPE/Embedded Floating-point Unavailable Exception

The SPE/embedded floating-point unavailable exception is taken if MSR[SPE] is cleared and execution of a SPE instruction other than the **brinc** instruction is attempted. When the SPE/embedded floating-point unavailable exception occurs, the processor suppresses execution of the instruction causing the exception. The SRR0, SRR1, MSR, and ESR registers are modified as follows:

- SRR0 is set to the effective address of the instruction causing the exception.
- SRR1 is set to the contents of the MSR at the time of the exception.
- MSR[CE, ME, DE] are unchanged. All other bits are cleared.
- ESR[SPE] is set. All other ESR bits are cleared.

Instruction execution resumes at address IVPR[0–15]||IVOR32[16–27]||0b0000.

20.3.8.3.2 SPE Vector Alignment Exception

For z7260n3, the SPE vector alignment exception is taken if the effective address of any of the following instructions is not aligned to a 32-bit boundary: **evldd[u]**, **evlddx**, **evldw[u]**, **evldwx**, **evldh[u]**, **evldhx**, **evstdd[u]**, **evstddx**, **evstdw[u]**, **evstdwx**, **evstdh[u]**, and **evstdhx**. When an SPE vector alignment exception occurs, the processor suppresses the execution of the instruction causing the alignment exception and takes an alignment interrupt.

SRR0, SRR1, MSR, ESR, and DEAR are modified as follows:

- SRR0 is set to the effective address of the instruction causing the alignment exception.
- SRR1 is set to the contents of the MSR at the time of the exception.
- MSR[CE, ME, DE] are unchanged. All other bits are cleared.
- ESR[SPE] (bit 24) is set. ESR[ST] is set only if the instruction causing the exception is a store and is cleared for a load. All other bits are cleared.
- DEAR is updated with the effective address of a byte of the load or store.

Instruction execution resumes at address IVPR[0–15]||IVOR5[16–27]||0b0000.

20.3.8.4 Exception Priorities

The following list shows the priority order in which exceptions are taken:

1. SPE Unavailable exception
2. SPE Vector Alignment exception

An SPE vector alignment exception is taken if an SPE double-word vector load or store access is attempted with an address which is not 32-bit aligned.

20.3.9 SPE Instruction Timing

Instruction timing in number of processor clock cycles for SPE instructions are shown in the following tables. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during divide execution.

20.3.9.1 SPE Simple Vector Arithmetic Instructions Timing

The following table shows instruction timing for SPE integer simple instructions. The table is sorted by opcode. These instructions are issued as a pair of operations.

Table 20-14. Simple Vector Arithmetic Instruction Timing

Basic Operation	Instruction	Latency	Throughput
Absolute Value	evabsb, evabsh, evabs, evabsd	1	1
	evabsbs, evabshs, evabss, evabsds	1	1
Absolute Difference	evabsdifsb, evabsdifsh, evabsdifsw, evabsdifub, evabsdifuh, evabsdifuw	1	1
Add	evaddb, evaddh, evaddw, evaddd	1	1
	evaddbss, evaddhss, evaddwss, evadddss evaddbus, evaddhus, evaddwus, evadddus	1	1
	evaddhx, evaddhxss, evaddhxus	1	1
	evaddwx, evaddwxss, evaddwxus	1	1
	evaddib, evaddih, evaddiw	1	1
	evaddsmiaaw, evaddssiaaw, evaddumiaaw, evaddusiaaw	1	1
	evaddsmiaa, evaddssiaa, evaddusiaa	1	1
AddSubf	evadd2subf2h, evadd2subf2hss	1	1
	evaddsubfh, evaddsubfhss	1	1
	evaddsubfhx, evaddsubfhxss	1	1
	evaddsubfw, evaddsubfwss	1	1
	evaddsubfwx, evaddsubfwxss	1	1
Average	evavgbs, evavghs, evavgws, evavgds, evavgbsr, evavghsr, evavgwsr, evavgdsr evavgbu, evavghu, evavgwu, evavgdu evavgbur, evavghur, evavgwur, evavgdur	1	1
Count Leading	evcntlsh, evcntlzh evcntlsw, evcntlzw	1	1
Extend	evextsb, evextzb	1	1
	evextsbh	1	1
	evextsh, *evextzh* (use evclrh)	1	1
	evextsw	1	1
Maximum	evmaxbs, evmaxhs, evmaxws, evmaxds evmaxbu, evmaxhu, evmaxwu, evmaxdu	1	1
	evmaxbpsh, evmaxbpuh	1	1
	evmaxhpsw, evmaxhpuw	1	1
	evmaxwpsd, evmaxwpud	1	1
Maximum Magnitude	evmaxmagws	1	1

Table continues on the next page...

Table 20-14. Simple Vector Arithmetic Instruction Timing (continued)

Basic Operation	Instruction	Latency	Throughput
Minimum	evminbs, evminhs, evminws, evminds evminbu, evminhu, evminwu, evmindu	1	1
	evminbpsh, evminbpuh	1	1
	evminhpsw, evminhpuw	1	1
	evminwpsd, evminwpud	1	1
Negate	evnegb, evnegh, evneg, evnegd	1	1
	evnegbs, evneghs, evnegs, evnegds	1	1
	evnegbo, evnegho, evnegwo	1	1
	evnegbos, evneghos, evnegwos	1	1
Round	evrndhb, evrndhbss, evrndhbus	1	1
	evrndhnb, evrndhnbss, evrndhnbus	1	1
	evrndwh, evrndwhss, evrndwhus	1	1
	evrndwnh, evrndwnhss, evrndwnhus	1	1
	evrnddw, evrnddwss, evrnddwus	1	1
	evrndndw, evrndndwss, evrndndwus	1	1
Sum of Absolute Differences	evsad2sh, evsad2sha, evsad2shaaw	1	1
	evsad2uh, evsad2uha, evsad2uhaaw	1	1
	evsad4sb, evsad4sba, evsad4sbaaw	1	1
	evsad4ub, evsad4uba, evsad4ubaaw	1	1
	evsadsw, evsadswa, evsadswaa	1	1
	evsaduw, evsaduwa, evsaduwaa	1	1
Saturate	evsatsbub	1	1
	evsatubsb	1	1
	evsatsdsw, evsatsduw	1	1
	evsatuduw	1	1
	evsatshsb, evsatshub	1	1
	evsatshuh	1	1
	evsatuhub	1	1
	evsatuhsh	1	1
	evsatswgsdf	1	1
	evsatswsh, evsatswuh	1	1
	evsatswuw	1	1
	evsatuwuh	1	1
	evsatuwsw	1	1
Subf	evsubfb, evsubfh, evsubfw, evsubfd	1	1
	evsubfbss, evsubfhss, evsubfwss, evsubfdss evsubfbus, evsubfhus, evsubfwus, evsubfdus	1	1
	evsubfhx, evsubfhxss, evsubfhxus	1	1
	evsubfwx, evsubfwxss, evsubfwxus	1	1

Table continues on the next page...

Table 20-14. Simple Vector Arithmetic Instruction Timing (continued)

Basic Operation	Instruction	Latency	Throughput
	evsubifb, evsubifh, evsubifw	1	1
	evsubfsmiaaw, evsubfssiaaw, evsubfumiaaw, evsubfusiaaw	1	1
	evsubfsmiaa, evsubfssiaa, evsubfusiaa	1	1
SubfAdd	evsubf2add2h, evsubf2add2hss	1	1
	evsubfaddh, evsubfaddhss	1	1
	evsubfaddhx, evsubfaddhxss	1	1
	evsubfaddw, evsubfaddwss	1	1
	evsubfaddwx, evsubfaddwxss	1	1
Summation/ Diff	evsumws, evsumwu, evsumwsa, evsumwua	1	1
	evsumwsaa, evsumwuaa	1	1
	evsum2hs, evsum2hu, evsum2hsa, evsum2hua	1	1
	evsum2hsaaw, evsum2huaaw	1	1
	evsum4bs, evsum4bu, evsum4bsa, evsum4bua	1	1
	evsum4bsaaw, evsum4buaaw	1	1
	evsum2his, evsum2hisa	1	1
	evsum2hisaaw	1	1
	evdiff2his, evdiff2hisa	1	1
	evdiff2hisaaw	1	1

20.3.9.2 SPE Complex Integer Instruction Timing

The following table shows instruction timing for SPE complex integer instructions. For the divide instructions, the number of stall cycles is (latency) for following instructions.

Table 20-15. SPE Complex Integer Instruction Timing

Operation	Instruction	Latency	Throughput
Divide	evdivws, evdivwu, evdivwsf, evdivwuf evdivs, evdivu	12-32	12-32[1]

1. Timing is data dependent

20.3.9.3 SPE Vector Logical Instruction Timing

The following table shows instruction timing for SPE simple vector logical instructions.

Table 20-16. SPE Vector Logical Instruction Timing

Basic Operation	Instruction	Latency	Throughput
AND	evand	1	1
ANDC	evandc	1	1
Clear	evclrbe, evclrbo	1	1
	evclrh	1	1
NAND	evnand	1	1
NOR	evnor	1	1
OR	evor	1	1
ORC	evorc	1	1
XNOR	eveqv	1	1
XOR	evxor	1	1

20.3.9.4 SPE Vector Shift/Rotate Instruction Timing

Instruction timing for SPE vector shift/rotate instructions is shown in the following table.

Table 20-17. SPE Vector Shift/Rotate Instruction Timing

Basic Operation	Instruction	Latency	Throughput
Shift Left	evslb, evslh, evslw, evsl evslbi, evslhi, evslwi, evsli	1	1
	evsloi	1	1
Logical Shift Right	evsrbu, evsrhu, evsrwu, evsru evsrbiu, evsrhiu, evsrwiu, evsriu	1	1
	evsroiu	1	1
Arithmetic Shift Right	evsrbs, evsrhs, evsrws, evsrs evsrbis, evsrhis, evsrwis, evsris	1	1
	evsrois	1	1
Rotate Left	evrlb, evrlh, evrlw evrlbi, evrlhi, evrlwi	1	1

20.3.9.5 SPE Vector Compare and Vector Set Instruction Timing

Instruction timing for SPE vector compare and set instructions is shown in Table 20-18 and Table 20-19. Table 20-18 shows the SPE vector compare instruction timing.

Table 20-18. SPE Vector Compare Instruction Timing

Basic Comparison Operation	Instruction	Latency	Throughput
=	evcmpeq, evcmpeqd	1	1
>	evcmpgts, evcmpgtu, evcmpgtds, evcmpgtdu	1	1
<	evcmplts, evcmpltu, evcmpltds, evcmpltdu	1	1

Table 20-19 shows the SPE vector set instruction timing.

Table 20-19. SPE Vector Set Instruction Timing

Comparison Operation	Instruction	Latency	Throughput
=	evseteqb[.], evseteqh[.], evseteqw[.]	1	1
>	evsetgtbs[.], evsetgtbu[.], evsetgths[.], evsetgthu[.], evsetgtws[.], evsetgtwu[.]	1	1
<	evsetltbs[.], evsetltbu[.], evsetlths[.], evsetlthu[.], evsetltws[.], evsetltwu[.]	1	1

20.3.9.6 SPE Vector Select Instruction Timing

The following table shows instruction timing for SPE vector select instructions.

Table 20-20. SPE Vector Select Instruction Timing

Operation	Instruction	Latency	Throughput
Select	evsel	1	1
Select Bits	evselbit	1	1
	evselbitm0	1	1
	evselbitm1	1	1

20.3.9.7 SPE Vector Data Arrangement Instruction Timing

The following table shows the instruction timing for SPE vector data arrangement instructions.

Table 20-21. SPE Vector Data Arrangement Instruction Timing

Operation	Instruction	Latency	Throughput
De-interleave	evdlveb	1	1
	evdlveob	1	1
	evdlvob	1	1
	evdlvoeb	1	1
	evdlveh	1	1
	evdlveoh	1	1
	evdlvoh	1	1
	evdlvoeh	1	1
Interleave	evilveh	1	1
	evilveoh	1	1
	evilvhih	1	1
	evilvhiloh	1	1
	evilvloh	1	1
	evilvlohih	1	1
	evilvoeh	1	1
	evilvoh	1	1
Merge	evmergehi	1	1
	evmergehilo	1	1
	evmergelo	1	1
	evmergelohi	1	1
Permute	evperm	1	1
	evperm2	1	1
	evperm3	1	1
Pack	evpksdsws, evpkuduws	1	1
	evpksdswfrs	1	1
	evpksdshefrs	1	1
	evpkshsbs, evpkshubs, evpkuhubs	1	1
	evpkswgshefrs	1	1
	evpkswgswfrs	1	1
	evpkswshs, evpkswuhs, evpkuwuhs	1	1
	evpkswshilvs	1	1
	evpkswshfrs	1	1
	evpkswshilvfrs	1	1
Splat	evsplatb	1	1
	evsplath	1	1
	evsplatfib, splatfih, splatfi	1	1
	evsplatfiba, splatfiha, splatfia	1	1
	evsplatfid	1	1
	evsplatfida	1	1

Table continues on the next page...

Table 20-21. SPE Vector Data Arrangement Instruction Timing (continued)

Operation	Instruction	Latency	Throughput
	evsplatfibo, splatfiho, splatfio	1	1
	evsplatfiboa, splatfihoa, splatfioa	1	1
	evsplatib, evsplatih, evsplati	1	1
	evsplatiba, evsplatiha, evsplatia	1	1
	evsplatid	1	1
	evsplatida	1	1
	evsplatibe, evsplatihe, evsplatie	1	1
	evsplatibea, evsplatihea, evsplatiea	1	1
Swap	evswapbhilo	1	1
	evswapblohi	1	1
	evswaphe	1	1
	evswaphhi	1	1
	evswaphhilo	1	1
	evswaphlo	1	1
	evswaphlohi	1	1
	evswapho	1	1
Unpack	evunpkhibsi, evunpkhibui, evunpklobsi, evunpklobui	1	1
	evunpkhihf, evunpkhihsi, evunkpkhihui, evunpklohf, evunpklohsi, evunkpklohui	1	1
	evunpkhiwgsf, evunpklowgsf	1	1
Extract	evxtrb	1	1
	evxtrd	1	1
	evxtrh	1	1
Insert	evinsb	1	1
	evinsh	1	1

20.3.9.8 SPE Multiply and Multiply/Accumulate Instruction Timing

The following table shows instruction timing for SPE multiply and multiply/accumulate instructions.

Table 20-22. SPE Multiply and Multiply/Accumulate Instruction Timing

Instruction	Latency	Throughput
all evm{b,h,w} instructions	4	1

20.3.9.9 SPE Dot Product Instruction Timing

The following table shows instruction timing for SPE dot product instructions.

Table 20-23. SPE Dot Product Instruction Timing

Instruction	Latency	Throughput
all evdotp instructions	4	1

20.3.9.10 SPE Misc. Vector Instruction Timing

The following table shows instruction timing for SPE miscellaneous instructions.

Table 20-24. SPE Misc. Vector Instruction Timing

Operation	Instruction	Latency	Throughput
load vector for shift	evlvsl	1	1
	evlvsr	1	1
store Accumulator	evmar	1	1
load Accumulator	evmra	1	1
Bit reversed increment	brinc	1	1

20.3.9.11 SPE Load and Store Instruction Timing

The following table shows instruction timing for SPE load and store instructions.

Table 20-25. SPE Load and Store Instruction Timing

Instruction	Latency	Throughput
all ev loads	3	1
all ev stores	3	1

Chapter 21
Crossbar Switch (AXBS)

21.1 Chip-specific AXBS information

21.1.1 Crossbar switch configuration

NOTE

For logical bus master assignments refer SMPU_x Logical bus Master Assignments.

This device contains three crossbar switch modules - Data Crossbar (AXBS_0), Instruction Crossbar (AXBS_1) and PRAM XBAR. The following figure illustrates that AXBS_0 and AXBS_1 are connected via a cross-lake, which is composed of intelligent bridging bus gaskets. These bus gaskets handle the traffic that crosses between the crossbars and do not require any interaction with the user. This dual-crossbar architecture allows all masters to access all slaves.

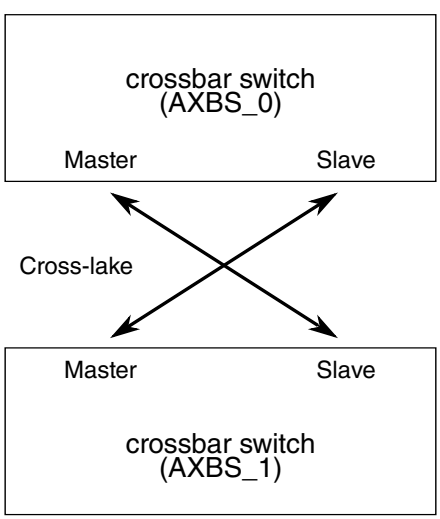


Figure 21-1. Crossbar switch configuration

For details about PRAM XBAR, please refer PRAM XBAR Overview section.

21.1.1.1 AXBS_0 configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

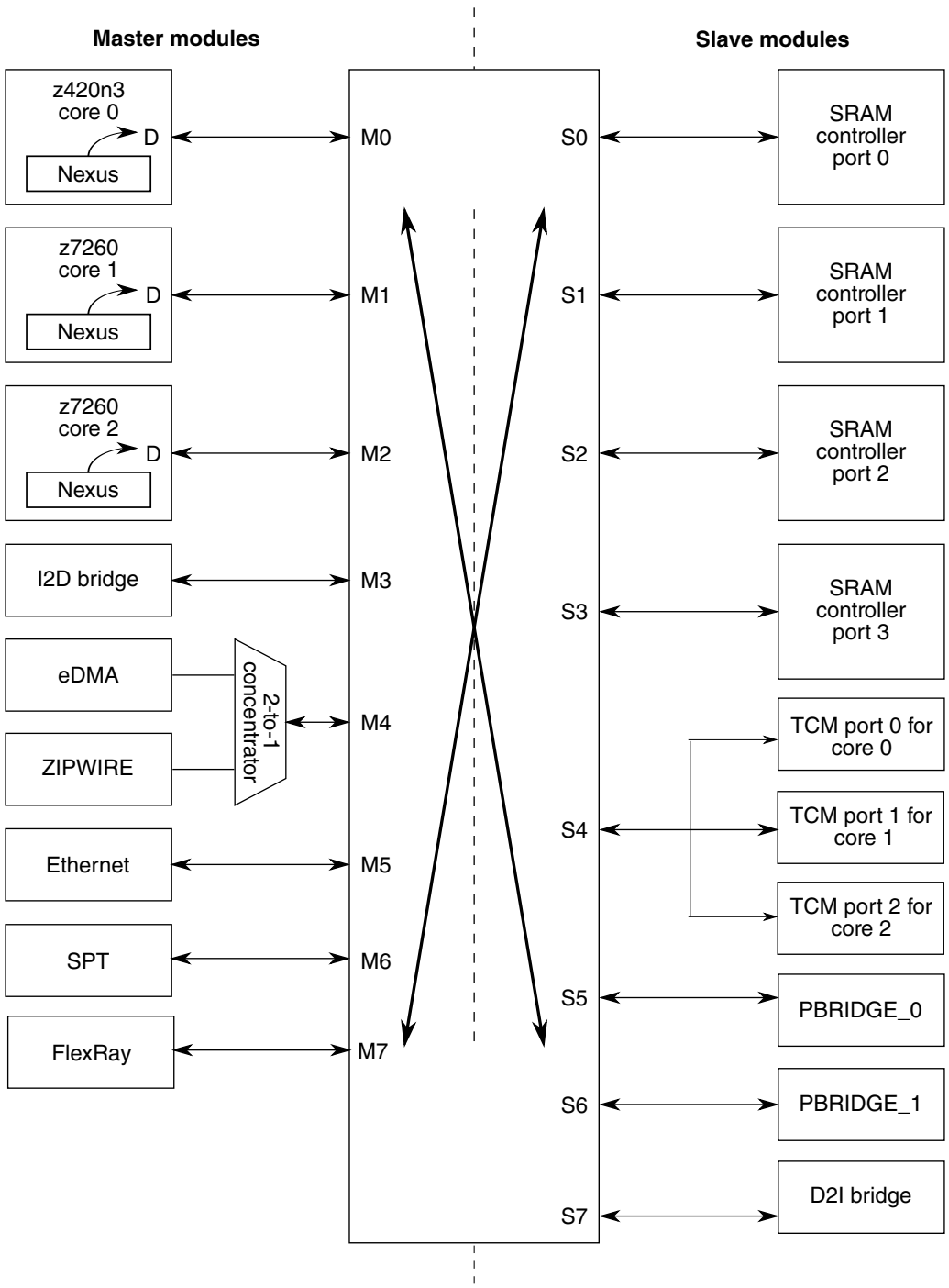


Figure 21-2. AXBS_0 Configuration

NOTE

Z4 TCM should not be accessed by any master while it is in STOP mode. No TCM should be accessed by any master if the corresponding core is still being held in reset.

21.1.1.1.1 AXBS_0 master assignments

This device contains eight masters connected to AXBS_0. The master assignments are shown in the table below:

Table 21-1. AXBS_0 master assignments

Master module	Physical master port number
Core0 data bus and Core0 Nexus	0
Core1 data bus and Core1 Nexus	1
Core2 data bus and Core2 Nexus	2
Instruction-to-Data (I2D) bus bridge	3
eDMA, Zipwire (merged via port concentrator)	4
Ethernet	5
Signal Processing Toolbox (SPT)	6
FlexRay	7

21.1.1.1.2 AXBS_0 slave assignments

This device contains eight slaves connected to AXBS_0. The slave assignments are shown in the table below:

Table 21-2. AXBS_0 slave assignments

Slave Module	Physical slave number
SRAM controller port for accesses from Core0/ CSE/ eDMA/ Zipwire/ Ethernet/FlexRay	0
SRAM controller port 1 for accesses from Core1	1
SRAM controller port 2 for accesses from Core2	2
SRAM controller port 3 for accesses from SPT [1]	3
TCM backdoor ports	4
PBRIDGE_0	5
PBRIDGE_1	6
Data-to-Instruction (D2I) bus bridge	7

1. SPT performance is optimized with a dedicated AXBS slave port. Since it is a direct master to slave port, arbitration and AXBS configuration of that slave will have no effect.

21.1.1.1.3 AXBS_0 High Priority Elevation

This device contains masters connected to AXBS_0. The master assignments are shown in the table below:

Table 21-3. AXBS_0 High Priority Elevation Feature

Master Module	Physical Master Port Number	High Priority Elevation	Additional Programming
Core0 data bus	0	x	In addition to setting HPE0 to 1 in AXBS_CRSn register for the applicable slave , set to 1 either or both of these fields in the special-purpose HID1 register for Core0: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Core1 data bus	1	x	In addition to setting HPE1 to 1 in AXBS_CRSn register for the applicable slave , set to 1 either or both of these fields in the special-purpose HID1 register for Core1: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Core2 data bus	2	x	In addition to setting HPE2 to 1 in AXBS_CRSn register for the applicable slave , set to 1 either or both of these fields in the special-purpose HID1 register for Core2: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Instruction-to-Data (I2D) bus bridge	3	x	High priority access will pass from Instruction Crossbar to Data crossbar.
eDMA/Zipwire	4	x[1]	For eDMA , in addition to setting HPE4 bit to 1 , BWC field in the eDMA_TCDn_CSR also needs to be configured for 1.
Ethernet	5	-	-

Table continues on the next page...

Table 21-3. AXBS_0 High Priority Elevation Feature (continued)

Signal Processing Toolbox (SPT)	6	-	-
FlexRay	7	x	No additional programming needed other than setting HPE7 bit for the applicable slave, to 1.

1. Zipwire doesn't support High Priority Elevation feature

21.1.1.1.4 AXBS_0 register reset values

All reset values for the AXBS_0 registers that are device specific are shown in the following table.

Table 21-4. AXBS_0 reset values

Register	Reset value
AXBS_0_PRSn	0x76543210
AXBS_0_CRSn	0x00100010

21.1.1.2 AXBS_1 configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

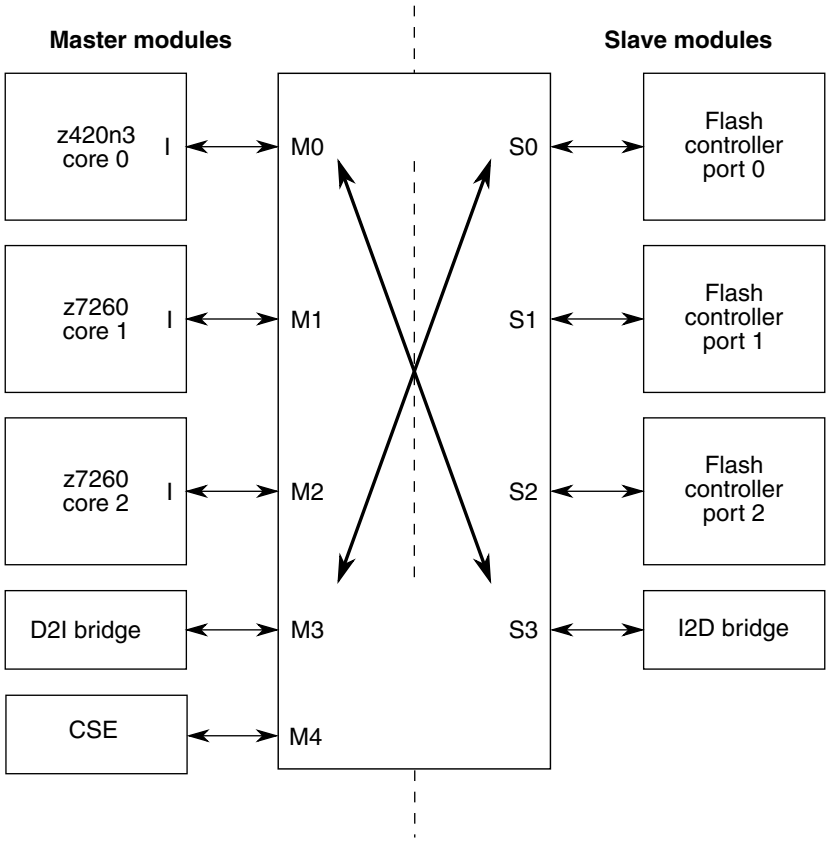


Figure 21-3. AXBS_1 Configuration

21.1.1.2.1 AXBS_1 master assignments

This device contains five masters connected to AXBS_1. The master assignments are shown in the table below:

NOTE

Slave ports 4-7 for AXBS_1 are not present, therefore AXBS_1_PRS4-7 registers and AXBS_1_CRS4-7 registers are 'reserved'.

Table 21-5. AXBS_1 master assignments

Master module	Physical master port number
Core0 instruction bus	0
Core1 instruction bus	1
Core2 instruction bus	2
Data-to-Instruction bus bridge.	3
CSE	4

21.1.1.2.2 AXBS_1 slave assignments

This device contains four slaves connected to AXBS_1. The slave assignments are shown in the table below:

Table 21-6. AXBS_1 slave assignments

Slave Module	Physical slave number
Flash controller port 0 for accesses from Core0 and all data-side masters (DMA, FlexRay, SPT, ENET)	0
Flash controller port 1 for accesses from Core1	1
Flash controller port 2 for accesses from Core2 and CSE	2
Instruction-to-Data (I2D) bus bridge	3

21.1.1.2.3 AXBS_1 High Priority Elevation

This device contains masters connected to AXBS_1. The master assignments are shown in the table below:

Table 21-7. AXBS_1 High Priority Elevation Feature

Master Module	Physical Master Port Number	High Priority Elevation	Additional Programming
Core0 instruction bus	0	x	In addition to setting HPE0 to 1 in AXBS_CRSn register for the applicable slave , set to 1 either or both of these fields in the special-purpose HID1 register for Core0: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Core1 instruction bus	1	x	In addition to setting HPE1 to 1 in AXBS_CRSn register for the applicable slave , set to 1 either or both of these fields in the special-purpose HID1 register for Core1: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Core2 instruction bus	2	x	In addition to setting HPE2 to 1 in AXBS_CRSn register for the applicable slave , set to 1

Table continues on the next page...

Table 21-7. AXBS_1 High Priority Elevation Feature (continued)

			either or both of these fields in the special-purpose HID1 register for Core2: • Elevate External Exceptions (EEE), bit 22 • Elevate Critical Exceptions (ECE), bit 23
Data-to-Instruction (D2I) bus bridge	3	x	High priority access will pass from Data crossbar to Instruction Crossbar.
CSE	4	-	-

21.1.1.2.4 AXBS_1 register reset values

All reset values for the AXBS_1 registers that are device specific are shown in the following table.

Table 21-8. AXBS_1 reset values

Register	Reset value
AXBS_1_PRSn	0x00043210
AXBS_1_CRSn	0x000F0010

21.2 Introduction

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

21.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation

  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave-by-slave basis
- Up to single-clock 64-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support
- Dynamic master priority elevation

21.3 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

AXBS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Registers Slave (AXBS_PRS0)	32	R/W	See section	21.3.1/896
10	Control Register (AXBS_CRS0)	32	R/W	See section	21.3.2/899
100	Priority Registers Slave (AXBS_PRS1)	32	R/W	See section	21.3.1/896
110	Control Register (AXBS_CRS1)	32	R/W	See section	21.3.2/899

Table continues on the next page...

AXBS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	Priority Registers Slave (AXBS_PRS2)	32	R/W	See section	21.3.1/896
210	Control Register (AXBS_CRS2)	32	R/W	See section	21.3.2/899
300	Priority Registers Slave (AXBS_PRS3)	32	R/W	See section	21.3.1/896
310	Control Register (AXBS_CRS3)	32	R/W	See section	21.3.2/899
400	Priority Registers Slave (AXBS_PRS4)	32	R/W	See section	21.3.1/896
410	Control Register (AXBS_CRS4)	32	R/W	See section	21.3.2/899
500	Priority Registers Slave (AXBS_PRS5)	32	R/W	See section	21.3.1/896
510	Control Register (AXBS_CRS5)	32	R/W	See section	21.3.2/899
600	Priority Registers Slave (AXBS_PRS6)	32	R/W	See section	21.3.1/896
610	Control Register (AXBS_CRS6)	32	R/W	See section	21.3.2/899
700	Priority Registers Slave (AXBS_PRS7)	32	R/W	See section	21.3.1/896
710	Control Register (AXBS_CRS7)	32	R/W	See section	21.3.2/899

21.3.1 Priority Registers Slave (AXBS_PRS*n*)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

NOTE

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains fewer than five masters, only two bits are valid.
- If five or more masters are present, all three bits of the priority field are used.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	M7			0	M6			0	M5			0	M4		
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

AXBS_PRS*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 M7	Master 7 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 M6	Master 6 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 M5	Master 5 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port.

Table continues on the next page...

AXBS_PRS*n* field descriptions (continued)

Field	Description
	110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 M4	Master 4 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 M3	Master 3 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port.

Table continues on the next page...

AXBS_PRS*n* field descriptions (continued)

Field	Description
	101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

21.3.2 Control Register (AXBS_CRS*n*)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Not all HPE*n* fields may be active. See the chip-specific crossbar information for which masters support high priority elevation. Setting a field corresponding to a master that does not support high-priority elevation has no effect.

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HLP	0						HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	1	0*	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

AXBS_CRS*n* field descriptions

Field	Description
0 RO	Read Only Forces the PRS*n* and CRS*n* registers to be read-only. After being set, only a hardware reset clears this field. 0 The CRS*n* and PRS*n* registers are writeable 1 The CRS*n* and PRS*n* registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HLP	Halt Low Priority Sets the initial arbitration priority for low power mode requests . Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports. 0 The low power mode request has the highest priority for arbitration on this slave port 1 The low power mode request has the lowest initial priority for arbitration on this slave port
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 HPE7	On this slave port, enable priority elevation for master 7. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 7 is disabled on this slave port 1 Priority elevation for master 7 is enabled on this slave port
9 HPE6	On this slave port, enable priority elevation for master 6. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 6 is disabled on this slave port 1 Priority elevation for master 6 is enabled on this slave port
10 HPE5	On this slave port, enable priority elevation for master 5. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 5 is disabled on this slave port 1 Priority elevation for master 5 is enabled on this slave port
11 HPE4	On this slave port, enable priority elevation for master 4. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 4 is disabled on this slave port 1 Priority elevation for master 4 is enabled on this slave port
12 HPE3	On this slave port, enable priority elevation for master 3. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 3 is disabled on this slave port 1 Priority elevation for master 3 is enabled on this slave port
13 HPE2	On this slave port, enable priority elevation for master 2. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 2 is disabled on this slave port 1 Priority elevation for master 2 is enabled on this slave port
14 HPE1	On this slave port, enable priority elevation for master 1. If enabled, the master is able to elevate its priority to the highest.

Table continues on the next page...

AXBS_CRS*n* field descriptions (continued)

Field	Description
	0 Priority elevation for master 1 is disabled on this slave port 1 Priority elevation for master 1 is enabled on this slave port
15 HPE0	On this slave port, enable priority elevation for master 0. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 0 is disabled on this slave port 1 Priority elevation for master 0 is enabled on this slave port
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 Low-power park. When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

21.4 Functional Description

21.4.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK] . This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

21.4.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

21.4.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

21.4.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 21-9. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true: • The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	At the next clock edge
Both of the following are true: • The current master is running a fixed length burst transfer or a locked transfer. • The requesting master's priority level is higher than that of the current master.	At the end of the burst transfer or locked transfer
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

21.4.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

21.4.3.3 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

21.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures that all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

NOTE

- During configuration of the crossbar switch, all other masters must be idle.
- To prevent reconfiguration of the crossbar switch, write 1 to CRS*n*[RO].

Chapter 22
Crossbar Integrity Checker (XBIC)

22.1 Chip-specific XBIC information

There are 5 instances of XBIC in this chip.

Table 22-1. XBIC configuration

Instance	Available on cross-bar	Master/Slave assignments
XBIC_1	AXBS_1 (instruction)	See AXBS_1 configuration
XBIC_0	AXBS_0 (data)	See AXBS_0 configuration
XBIC_2	PRAM AXBS	See PRAM_XBAR master assignments See PRAM_XBAR slave assignments
XBIC_3	DMA/SIPI splitter AXBS	Master: DMA, SIPI Slave: Data AXBS Master M4
XBIC_4	TCM splitter AXBS	Master: Data AXBS Slave port 4 Slave: Z4 TCM, Z7a TCM, Z7b TCM

Please see memory map spreadsheet attached with this document for more details.

22.2 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers.

22.3 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors

- Verification of feedback information for each data phase during crossbar transfer
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

22.4 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in the following diagram.

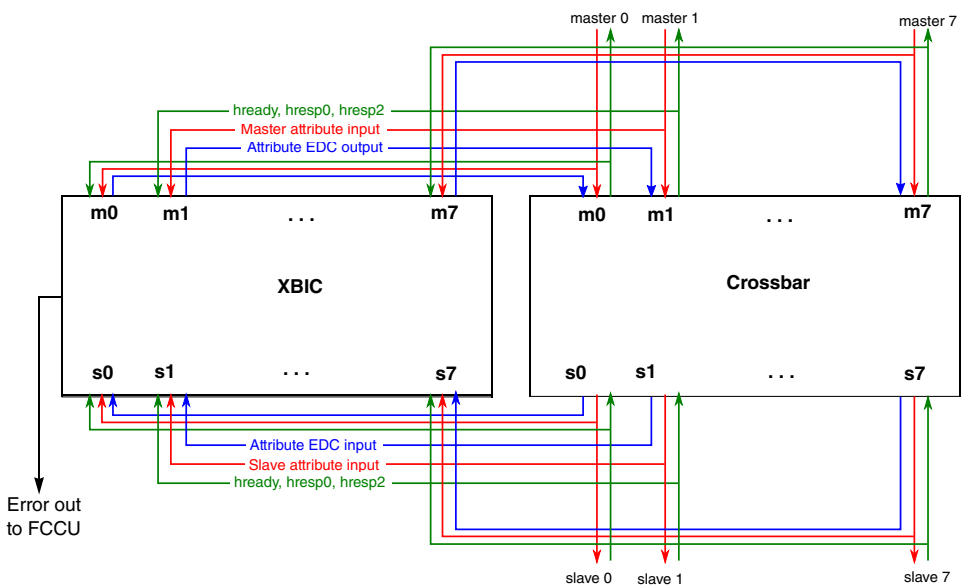


Figure 22-1. XBIC system block diagram

22.5 External signal description

The XBIC has no external interface signals.

22.6 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

XBIC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBIC Module Control Register (XBIC_MCR)	32	R/W	FFFF_0000h	22.6.1/909
4	XBIC Error Injection Register (XBIC_EIR)	32	R/W	0000_0000h	22.6.2/911
8	XBIC Error Status Register (XBIC_ESR)	32	R	0000_0000h	22.6.3/912
C	XBIC Error Address Register (XBIC_EAR)	32	R	0000_0000h	22.6.4/914

22.6.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 0. 1 Attribute integrity checking enabled for slave port 0.
1 SE1	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 1. 1 Attribute integrity checking enabled for slave port 1.

Table continues on the next page...

XBIC_MCR field descriptions (continued)

Field	Description
2 SE2	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 2. 1 Attribute integrity checking enabled for slave port 2.
3 SE3	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 3. 1 Attribute integrity checking enabled for slave port 3.
4 SE4	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 4. 1 Attribute integrity checking enabled for slave port 4.
5 SE5	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 5. 1 Attribute integrity checking enabled for slave port 5.
6 SE6	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 6. 1 Attribute integrity checking enabled for slave port 6.
7 SE7	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 7. 1 Attribute integrity checking enabled for slave port 7.
8 ME0	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 0. 1 Attribute integrity checking enabled for master port 0.
9 ME1	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 1. 1 Attribute integrity checking enabled for master port 1.
10 ME2	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 2. 1 Attribute integrity checking enabled for master port 2.
11 ME3	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 3. 1 Attribute integrity checking enabled for master port 3.
12 ME4	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 4. 1 Attribute integrity checking enabled for master port 4.
13 ME5	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 5. 1 Attribute integrity checking enabled for master port 5.

Table continues on the next page...

XBIC_MCR field descriptions (continued)

Field	Description
14 ME6	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 6. 1 Attribute integrity checking enabled for master port 6.
15 ME7	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 7. 1 Attribute integrity checking enabled for master port 7.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.6.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20–23 MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24–31 SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

22.6.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected, the slave port, the master port, and the syndrome are captured in the SLV, MST, and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are catured in the DPSE0-7 and DPME0-7 fields.

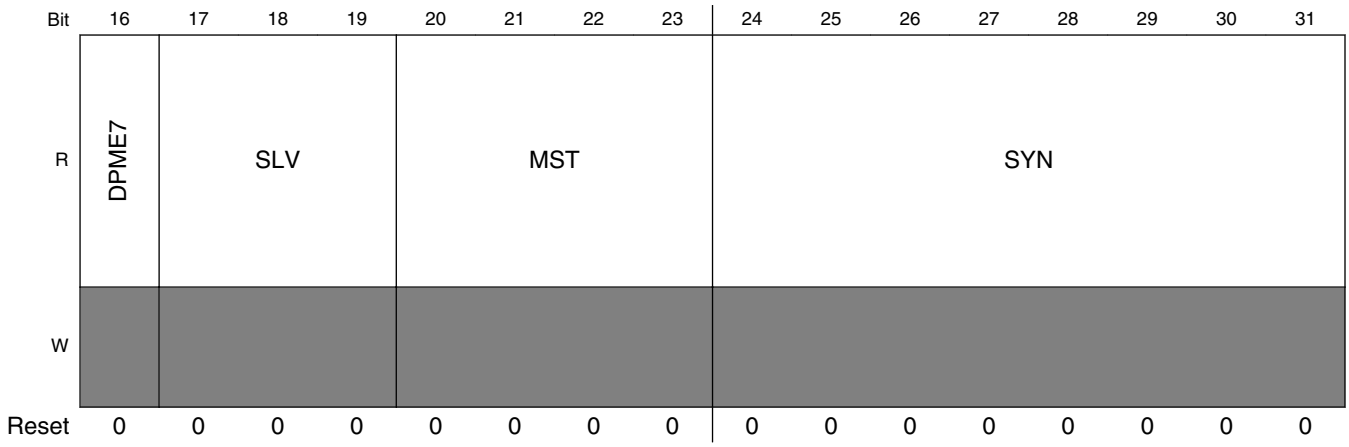
This register is cleared only on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_ESR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error. 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error. 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error. 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error. 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error. 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error. 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error. 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on slave port 7 1 Data phase error on slave port 7
9 DPME0	Data phase master port error. 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error. 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error. 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error. 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error. 0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error. 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error. 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error. 0 No error on master port 7 1 Data phase error on master port 7
17–19 SLV	Slave Port. The slave port targeted by the last transfer with an error.
20–23 MST	Master ID. The master ID value of the last transfer with an error.
24–31 SYN	Syndrome. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 22-2 .

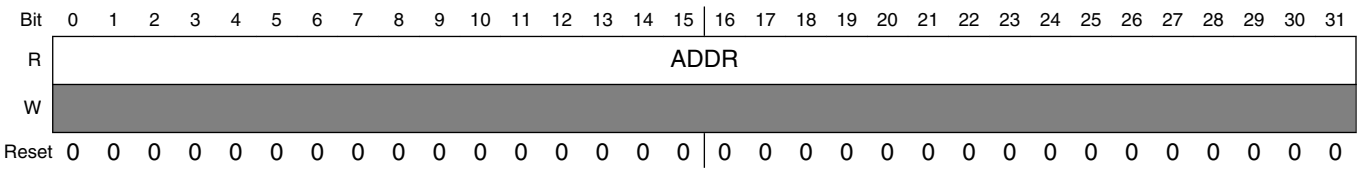
22.6.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the first transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_EAR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + Ch offset = Ch



XBIC_EAR field descriptions

Field	Description
0–31 ADDR	The address of the last transfer with an error.

22.7 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar interface. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a individual port basis. When an error is detected, it is captured in XBIC_ESR register and XBIC also sends out an error signal to the Fault Collection and Control Unit (FCCU). Detection of an error does not generate a bus error. The XBIC integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

During the AHB address phase, the crossbar attribute information is checked using an 8-bit Error Detection Code (EDC), which detects any single- or double-bit errors. When an error is detected, due either to hardware fault or error injection, information related to the error is captured and reported in XBIC_ESR and XBIC_EAR. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in Table 22-2. Any other syndrome indicates a multi-bit error.

During the AHB data phase, the responding signals from the slave to master are checked when passing through the crossbar. When an error is detected, the information is captured in the XBIC_ESR DPSE0-7 and DPME0-7 fields.

The XBIC can also be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared

Table 22-2. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

Chapter 23
Peripheral Bridge (AIPS-Lite)

23.1 Chip-specific AIPS information

23.1.1 Number of peripheral bridges

This device contains two identical peripheral bridges.

23.1.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See Memory Map chapter for the memory slot assignment for each module.

23.1.3 MPR registers

Each of the two peripheral bridges supports a maximum of sixteen crossbar switch masters, each assigned to MTR*n*,MTW*n* and MPL*n* field in the MPRA and MPRB registers. However, fewer are supported on this device. See Chip-specific AXBS information for details of the master port assignments for this device.

See Chip-specific SMPU information for details of the Logical bus Master Assignments for this device.

The reset value for MPRA is 0x7700_0007 in both bridges.

The reset value for MPRB is 0x7700_5007 in both bridges.

NOTE

In MPRB register , the MTR12, MTW12 & MPL12 bit fields belongs to CSE. By default MTR12 = 1 & MPL12 = 1 and these values cannot be changed by software. Only MTW12 value can be changed to 0/1.

23.1.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each are assigned to a PACR*n* or OPACR*n* field within the PACR A-H and OPACR A-AF registers. See memory map spreadsheet attached with this document for details of the peripheral slot assignments for this device. Unused PACR*n* fields are reserved.

The reset values for the OPACR registers are given in the following table.

Table 23-1. OPACR reset values

Registers	AIPS0	AIPS1
OPACRA	0x4444_4445	0x4444_4444
OPACRB	0x4444_4404	0x4444_4444
OPACRC	0x4444_4444	0x4444_4444
OPACRD	0x4444_4444	0x4444_4444
OPACRE	0x4444_4444	0x4444_4444
OPACRF	0x4444_4444	0x4444_4444
OPACRG	0x4444_4444	0x4444_4444
OPACRH	0x4444_4444	0x4444_4444
OPACRI	0x4444_4444	0x4444_4444
OPACRJ	0x4444_4444	0x4444_4444
OPACRK	0x4444_4444	0x4444_4444
OPACRL	0x4444_4444	0x4444_4444
OPACRM	0x4444_4444	0x4444_4444
OPACRN	0x4444_4444	0x4444_4444
OPACRO	0x4444_4444	0x4444_4444
OPACRP	0x4444_4444	0x4444_4444
OPACRQ	0x4444_4444	0x4444_4444
OPACRR	0x4444_4444	0x4444_4444
OPACRS	0x4444_4444	0x4444_4444
OPACRT	0x4444_4444	0x4444_4444
OPACRU	0x4444_4444	0x4444_4444
OPACRV	0x4444_4444	0x4444_4444
OPACRW	0x4444_4444	0x4444_4444
OPACRX	0x4444_4444	0x4444_4444
OPACRY	0x4444_4444	0x4444_4444
OPACRZ	0x4444_4444	0x4444_4444
OPACRAA	0x4444_4444	0x4444_4444
OPACRAB	0x4444_4444	0x4444_4444
OPACRAC	0x4444_4444	0x4444_4444
OPACRAD	0x4444_4444	0x4444_4444

Table continues on the next page...

Table 23-1. OPACR reset values (continued)

Registers	AIPS0	AIPS1
OPACRAE	0x4444_4444	0x4444_4444
OPACRAF	0x4444_4444	0x4444_4444

The reset values for the PACR registers are given in the following table.

Table 23-2. PACR reset values

Register	AIPS0	AIPS1
PACRA	0x5444_4444	0x5000_0000
PACRB	0x4444_4404	0x0000_0000
PACRC	0x4400_4440	0x0000_0000
PACRD	0x0044_4444	0x0000_0000
PACRF	0x4000_4000	0x0000_0000
PACRG	0x0000_0000	0x0000_0000
PACRH	0x0000_0000	0x0000_0000

23.2 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

23.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Programming model provides memory protection functionality

23.2.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

23.3 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	23.3.1/922
4	Master Privilege Register B (AIPS_MPRB)	32	R/W	See section	23.3.2/926
100	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	4444_4444h	23.3.3/929
104	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	4444_4444h	23.3.3/929
108	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	4444_4444h	23.3.3/929
10C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	4444_4444h	23.3.3/929
110	Peripheral Access Control Register (AIPS_PACR_Reserved)	32	R/W	4444_4444h	23.3.3/929
114	Peripheral Access Control Register (AIPS_PACRF)	32	R/W	4444_4444h	23.3.3/929
118	Peripheral Access Control Register (AIPS_PACRG)	32	R/W	4444_4444h	23.3.3/929
11C	Peripheral Access Control Register (AIPS_PACRH)	32	R/W	4444_4444h	23.3.3/929
140	Off-Platform Peripheral Access Control Register (AIPS_OPACRA)	32	R/W	See section	23.3.4/934
144	Off-Platform Peripheral Access Control Register (AIPS_OPACRB)	32	R/W	See section	23.3.4/934
148	Off-Platform Peripheral Access Control Register (AIPS_OPACRC)	32	R/W	See section	23.3.4/934
14C	Off-Platform Peripheral Access Control Register (AIPS_OPACRD)	32	R/W	See section	23.3.4/934
150	Off-Platform Peripheral Access Control Register (AIPS_OPACRE)	32	R/W	See section	23.3.4/934
154	Off-Platform Peripheral Access Control Register (AIPS_OPACRF)	32	R/W	See section	23.3.4/934

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
158	Off-Platform Peripheral Access Control Register (AIPS_OPACRG)	32	R/W	See section	23.3.4/934
15C	Off-Platform Peripheral Access Control Register (AIPS_OPACRH)	32	R/W	See section	23.3.4/934
160	Off-Platform Peripheral Access Control Register (AIPS_OPACRI)	32	R/W	See section	23.3.4/934
164	Off-Platform Peripheral Access Control Register (AIPS_OPACRJ)	32	R/W	See section	23.3.4/934
168	Off-Platform Peripheral Access Control Register (AIPS_OPACRK)	32	R/W	See section	23.3.4/934
16C	Off-Platform Peripheral Access Control Register (AIPS_OPACRL)	32	R/W	See section	23.3.4/934
170	Off-Platform Peripheral Access Control Register (AIPS_OPACRM)	32	R/W	See section	23.3.4/934
174	Off-Platform Peripheral Access Control Register (AIPS_OPACRN)	32	R/W	See section	23.3.4/934
178	Off-Platform Peripheral Access Control Register (AIPS_OPACRO)	32	R/W	See section	23.3.4/934
17C	Off-Platform Peripheral Access Control Register (AIPS_OPACRP)	32	R/W	See section	23.3.4/934
180	Off-Platform Peripheral Access Control Register (AIPS_OPACRQ)	32	R/W	See section	23.3.4/934
184	Off-Platform Peripheral Access Control Register (AIPS_OPACRR)	32	R/W	See section	23.3.4/934
188	Off-Platform Peripheral Access Control Register (AIPS_OPACRS)	32	R/W	See section	23.3.4/934
18C	Off-Platform Peripheral Access Control Register (AIPS_OPACRT)	32	R/W	See section	23.3.4/934
190	Off-Platform Peripheral Access Control Register (AIPS_OPACRU)	32	R/W	See section	23.3.4/934
194	Off-Platform Peripheral Access Control Register (AIPS_OPACRV)	32	R/W	See section	23.3.4/934
198	Off-Platform Peripheral Access Control Register (AIPS_OPACRW)	32	R/W	See section	23.3.4/934
19C	Off-Platform Peripheral Access Control Register (AIPS_OPACRX)	32	R/W	See section	23.3.4/934
1A0	Off-Platform Peripheral Access Control Register (AIPS_OPACRY)	32	R/W	See section	23.3.4/934
1A4	Off-Platform Peripheral Access Control Register (AIPS_OPACRZ)	32	R/W	See section	23.3.4/934
1A8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAA)	32	R/W	See section	23.3.4/934
1AC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAB)	32	R/W	See section	23.3.4/934

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1B0	Off-Platform Peripheral Access Control Register (AIPS_OPACRAC)	32	R/W	See section	23.3.4/934
1B4	Off-Platform Peripheral Access Control Register (AIPS_OPACRAD)	32	R/W	See section	23.3.4/934
1B8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAE)	32	R/W	See section	23.3.4/934
1BC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAF)	32	R/W	See section	23.3.4/934

23.3.1 Master Privilege Register A (AIPS_MPRA)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR0	MTW0	MPL0	0	MTR1	MTW1	MPL1	0	MTR2	MTW2	MPL2	0	MTR3	MTW3	MPL3
W																
Reset	0*	1*	1*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR4	MTW4	MPL4	0	MTR5	MTW5	MPL5	0	MTR6	MTW6	MPL6	0	MTR7	MTW7	MPL7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR0	Master 0 Trusted For Read

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR1	Master 1 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW1	Master 1 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL1	Master 1 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 MTR2	Master 2 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
10 MTW2	Master 2 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
11 MPL2	Master 2 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
14 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
15 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MTR5	Master 5 Trusted For Read Determines whether the master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
22 MTW5	Master 5 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
23 MPL5	Master 5 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MTR6	Master 6 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
26 MTW6	Master 6 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
27 MPL6	Master 6 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 MTR7	Master 7 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
30 MTW7	Master 7 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
31 MPL7	Master 7 privilege level Specifies how the privilege level of the master is determined.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.

23.3.2 Master Privilege Register B (AIPS_MPRB)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR8	MTW8	MPL8	0	MTR9	MTW9	MPL9	0	MTR10	MTW10	MPL10	0	Reserved		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR12	MTW12	MPL12	0	MTR13	MTW13	MPL13	0	MTR14	MTW14	MP14	0	MTR15	MTW15	MPL15
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRB field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR8	Master 8 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW8	Master 8 trusted for writes Determines whether the master is trusted for write accesses.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
	0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL8	Master 8 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR9	Master 9 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW9	Master 9 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL9	Master 9 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 MTR10	Master 10 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
10 MTW10	Master 10 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
11 MPL10	Master 10 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
13–15 Reserved	This field is reserved.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR12	Master 12 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW12	Master 12 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL12	Master 12 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MTR13	Master 13 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
22 MTW13	Master 13 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
23 MPL13	Master 13 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MTR14	Master 14 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
26 MTW14	Master 14 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
27 MP14	Master 14 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 MTR15	Master 15 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
30 MTW15	Master 15 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
31 MPL15	Master 15 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.

23.3.3 Peripheral Access Control Register (AIPS_PACR*n*)

Each of the on-platform peripherals has a 4-bit PACR*n* field, which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignment for each PACR field is defined by the memory map slot that the peripheral is assigned to. See the device's memory map details for the assignments for a particular device.

If a peripheral is absent, the corresponding PACR field is not implemented. Reads or writes to the location of an unimplemented PACR register (because all eight of its peripherals are absent) should be avoided to prevent undesired behavior.

The following table shows the top-level structure of PACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x10C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x11C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

NOTE

The reset value of PACR0 is 0101b.

Address: 0h base + 100h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPS_PACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

23.3.4 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)

Each of the on-platform peripherals has a 4-bit OPACR*n* field, which defines the access levels supported by the given module. Eight OPACR fields are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignment for each OPACR field is defined by the memory map slot that the peripheral is assigned to. See the device's memory map details for the assignments for a particular device.

If a peripheral is absent, the corresponding OPACR field is not implemented. Reads or writes to the location of an unimplemented OPACR register (because all eight of its peripherals are absent) should be avoided to prevent undesired behavior.

The following table shows the top-level structure of OPACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x140	OPACRA	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0x144	OPACRB	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0x148	OPACRC	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0x14C	OPACRD	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0x150	OPACRE	OPACR32	OPACR33	OPACR34	OPACR35	OPACR36	OPACR37	OPACR38	OPACR39
0x154	OPACRF	OPACR40	OPACR41	OPACR42	OPACR43	OPACR44	OPACR45	OPACR46	OPACR47
0x158	OPACRG	OPACR48	OPACR49	OPACR50	OPACR51	OPACR52	OPACR53	OPACR54	OPACR55
0x15C	OPACRH	OPACR56	OPACR57	OPACR58	OPACR59	OPACR60	OPACR61	OPACR62	OPACR63
0x160	OPACRI	OPACR64	OPACR65	OPACR66	OPACR67	OPACR68	OPACR69	OPACR70	OPACR71
0x164	OPACRJ	OPACR72	OPACR73	OPACR74	OPACR75	OPACR76	OPACR77	OPACR78	OPACR79
0x168	OPACRK	OPACR80	OPACR81	OPACR82	OPACR83	OPACR84	OPACR85	OPACR86	OPACR87
0x16C	OPACRL	OPACR88	OPACR89	OPACR90	OPACR91	OPACR92	OPACR93	OPACR94	OPACR95
0x170	OPACRM	OPACR96	OPACR97	OPACR98	OPACR99	OPACR100	OPACR101	OPACR102	OPACR103
0x174	OPACRN	OPACR104	OPACR105	OPACR106	OPACR107	OPACR108	OPACR109	OPACR110	OPACR111
0x178	OPACRO	OPACR112	OPACR113	OPACR114	OPACR115	OPACR116	OPACR117	OPACR118	OPACR119
0x17C	OPACRP	OPACR120	OPACR121	OPACR122	OPACR123	OPACR124	OPACR125	OPACR126	OPACR127
0x180	OPACRQ	OPACR128	OPACR129	OPACR130	OPACR131	OPACR132	OPACR133	OPACR134	OPACR135
0x184	OPACRR	OPACR136	OPACR137	OPACR138	OPACR139	OPACR140	OPACR141	OPACR142	OPACR143
0x188	OPACRS	OPACR144	OPACR145	OPACR146	OPACR147	OPACR148	OPACR149	OPACR150	OPACR151
0x18C	OPACRT	OPACR152	OPACR153	OPACR154	OPACR155	OPACR156	OPACR157	OPACR158	OPACR159
0x190	OPACRU	OPACR160	OPACR161	OPACR162	OPACR163	OPACR164	OPACR165	OPACR166	OPACR167
0x194	OPACRV	OPACR168	OPACR169	OPACR170	OPACR171	OPACR172	OPACR173	OPACR174	OPACR175
0x198	OPACRW	OPACR176	OPACR177	OPACR178	OPACR179	OPACR180	OPACR181	OPACR182	OPACR183
0x19C	OPACRX	OPACR184	OPACR185	OPACR186	OPACR187	OPACR188	OPACR189	OPACR190	OPACR191
0x1A0	OPACRY	OPACR192	OPACR193	OPACR194	OPACR195	OPACR196	OPACR197	OPACR198	OPACR199
0x1A4	OPACRZ	OPACR200	OPACR201	OPACR202	OPACR203	OPACR204	OPACR205	OPACR206	OPACR207
0x1A8	OPACRAA	OPACR208	OPACR209	OPACR210	OPACR211	OPACR212	OPACR213	OPACR214	OPACR215
0x1AC	OPACRAB	OPACR216	OPACR217	OPACR218	OPACR219	OPACR220	OPACR221	OPACR222	OPACR223
0x1B0	OPACRAC	OPACR224	OPACR225	OPACR226	OPACR227	OPACR228	OPACR229	OPACR230	OPACR231

Table continues on the next page...

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x1B4	OPACRAD	OPACR232	OPACR233	OPACR234	OPACR235	OPACR236	OPACR237	OPACR238	OPACR239
0x1B8	OPACRAE	OPACR240	OPACR241	OPACR242	OPACR243	OPACR244	OPACR245	OPACR246	OPACR247
0x1BC	OPACRAF	OPACR248	OPACR249	OPACR250	OPACR251	OPACR252	OPACR253	OPACR254	OPACR255

Address: 0h base + 140h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_OPACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute MPR *x* [MPL *n*], and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

23.4 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

23.4.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 24
System Memory Protection Unit (SMPU)

24.1 Chip-specific SMPU information

This chip contains two instances of the System Memory Protection Unit (SMPU):

- SMPU_0 for the data XBAR (XBAR_0)
- SMPU_1 for the instruction XBAR (XBAR_1)

The following sections explain how each SMPU is configured on this chip.

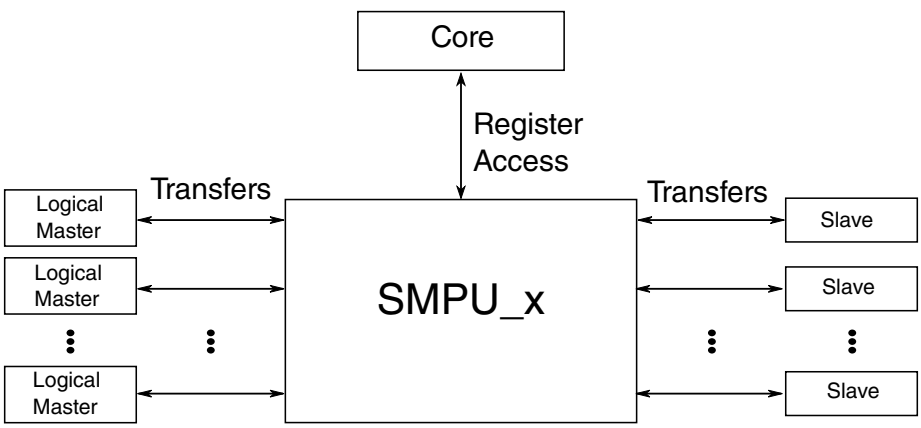


Figure 24-1. SMPU_x configuration

24.1.1 SMPU_x Logical bus Master Assignments

The logical bus master assignments for the SMPU_0/1 are shown in the following table.

Table 24-1. Master IDs

Module	Master ID
Core0 (Instruction and Data Bus)	0
Core1 (Instruction and Data Bus)	1
eDMA	2[1]

Table continues on the next page...

Table 24-1. Master IDs (continued)

Module	Master ID
FlexRay	3
Zipwire	4
ENET	5
SPT (PDMA)	6
Core2 (Instruction and Data Bus)	7
Core0 (Nexus)	8
Core1 (Nexus)	9
SPT (SDMA (MIPI_CSI image data and channel5))	10
CSE	12
SPT (Command Sequencer)	13
SPT (SDMA (ADC/MIPI_CSI))	14
Core2 (Nexus)	15

1. eDMA can emulate other master IDs, with exception of the CSE master ID

24.1.2 SMPU_EDRn[EATTR] fixed values for some masters

For each of the following masters, SMPU_EDRn[EATTR] always has the indicated fixed value:

- ENET : 01
- FlexRay : 11
- SPT : 11

24.2 Overview

The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors system bus transactions and evaluates their appropriateness using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

24.2.1 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The access signals coming from the crossbar switch slave ports are shown on the left. The access evaluation macros and region descriptors are in the middle, and the peripheral bus interface is on the right side.

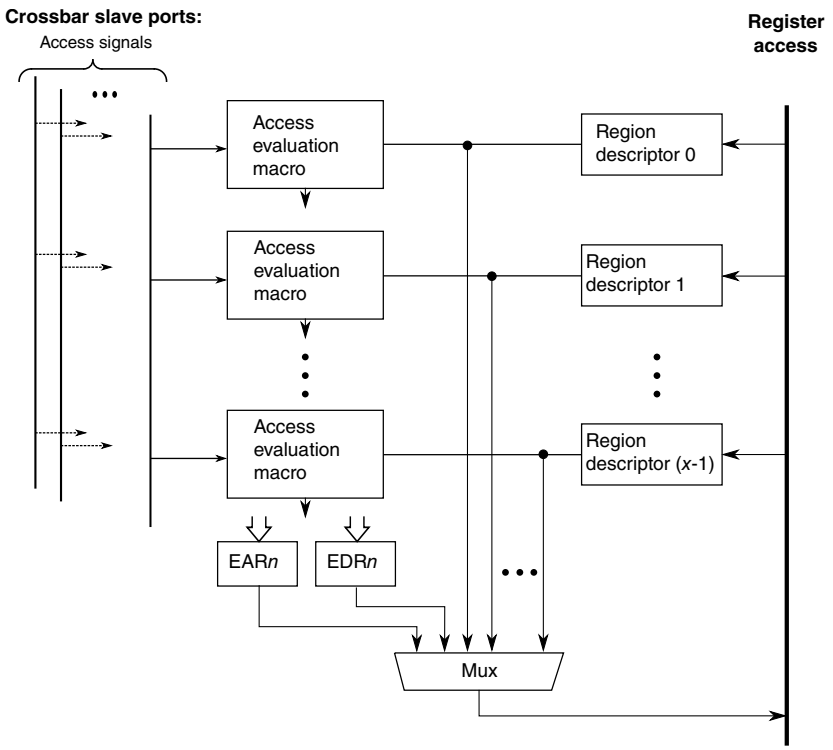


Figure 24-2. SMPU block diagram

The access evaluation macros determine hits to memory regions and detect violations to access protections. If a violation is detected, the macros update the appropriate master's error registers. For details of the access evaluation macro, see Access evaluation macro.

24.2.2 Features

The SMPU feature set includes:

- Arbitrarily sized protection regions alignable anywhere in memory.
  - Region sizes can vary from a minimum of 1 byte to a maximum of 4 GB.
- Read/write access control permissions are defined in the region descriptor.

- Global cache-inhibit attribute indicator.
- Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
- Priority given to granting permission over denying access for overlapping region descriptors.
- Detection of access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Instruction storage (ISI) or data storage (DSI) interrupts generated by aborted core accesses. Buffered writes generate Machine check exception (refer to the documentation related to the core for details).
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit.

24.3 Memory map/register definition

The programming model is partitioned into three groups:

- Control registers
- Error capture registers
- Data structure containing the region descriptors

The programming model can be referenced only in supervisor mode using 32-bit accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register, or a read of a write-only register), or in user mode generate an error termination.

NOTE

See the chip configuration details for any chip-specific register information for this module. For example, a specific instance of the module may support only eight or fewer region descriptors.

SMPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control/Error Status Register 0 (SMPU_CESR0)	32	R/W	0000_8002h	24.3.1/948

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Control/Error Status Register 1 (SMPU_CESR1)	32	R/W	0000_8004h	24.3.2/949
100	Error Address Register, Bus Master n (SMPU_EAR0)	32	R	0000_0000h	24.3.3/950
104	Error Detail Register, Bus Master n (SMPU_EDR0)	32	R	0000_0080h	24.3.4/951
108	Error Address Register, Bus Master n (SMPU_EAR1)	32	R	0000_0000h	24.3.3/950
10C	Error Detail Register, Bus Master n (SMPU_EDR1)	32	R	0000_0080h	24.3.4/951
110	Error Address Register, Bus Master n (SMPU_EAR2)	32	R	0000_0000h	24.3.3/950
114	Error Detail Register, Bus Master n (SMPU_EDR2)	32	R	0000_0080h	24.3.4/951
118	Error Address Register, Bus Master n (SMPU_EAR3)	32	R	0000_0000h	24.3.3/950
11C	Error Detail Register, Bus Master n (SMPU_EDR3)	32	R	0000_0080h	24.3.4/951
120	Error Address Register, Bus Master n (SMPU_EAR4)	32	R	0000_0000h	24.3.3/950
124	Error Detail Register, Bus Master n (SMPU_EDR4)	32	R	0000_0080h	24.3.4/951
128	Error Address Register, Bus Master n (SMPU_EAR5)	32	R	0000_0000h	24.3.3/950
12C	Error Detail Register, Bus Master n (SMPU_EDR5)	32	R	0000_0080h	24.3.4/951
130	Error Address Register, Bus Master n (SMPU_EAR6)	32	R	0000_0000h	24.3.3/950
134	Error Detail Register, Bus Master n (SMPU_EDR6)	32	R	0000_0080h	24.3.4/951
138	Error Address Register, Bus Master n (SMPU_EAR7)	32	R	0000_0000h	24.3.3/950
13C	Error Detail Register, Bus Master n (SMPU_EDR7)	32	R	0000_0080h	24.3.4/951
140	Error Address Register, Bus Master n (SMPU_EAR8)	32	R	0000_0000h	24.3.3/950
144	Error Detail Register, Bus Master n (SMPU_EDR8)	32	R	0000_0080h	24.3.4/951
148	Error Address Register, Bus Master n (SMPU_EAR9)	32	R	0000_0000h	24.3.3/950
14C	Error Detail Register, Bus Master n (SMPU_EDR9)	32	R	0000_0080h	24.3.4/951
150	Error Address Register, Bus Master n (SMPU_EAR10)	32	R	0000_0000h	24.3.3/950
154	Error Detail Register, Bus Master n (SMPU_EDR10)	32	R	0000_0080h	24.3.4/951
158	Error Address Register, Bus Master n (SMPU_EAR11)	32	R	0000_0000h	24.3.3/950
15C	Error Detail Register, Bus Master n (SMPU_EDR11)	32	R	0000_0080h	24.3.4/951
160	Error Address Register, Bus Master n (SMPU_EAR12)	32	R	0000_0000h	24.3.3/950
164	Error Detail Register, Bus Master n (SMPU_EDR12)	32	R	0000_0080h	24.3.4/951
168	Error Address Register, Bus Master n (SMPU_EAR13)	32	R	0000_0000h	24.3.3/950
16C	Error Detail Register, Bus Master n (SMPU_EDR13)	32	R	0000_0080h	24.3.4/951
170	Error Address Register, Bus Master n (SMPU_EAR14)	32	R	0000_0000h	24.3.3/950
174	Error Detail Register, Bus Master n (SMPU_EDR14)	32	R	0000_0080h	24.3.4/951
178	Error Address Register, Bus Master n (SMPU_EAR15)	32	R	0000_0000h	24.3.3/950
17C	Error Detail Register, Bus Master n (SMPU_EDR15)	32	R	0000_0080h	24.3.4/951
400	Region Descriptor n, Word 0 (SMPU_RGD0_WORD0)	32	R/W	0000_0000h	24.3.5/952
404	Region Descriptor n, Word 1 (SMPU_RGD0_WORD1)	32	R/W	0000_0000h	24.3.6/952
408	Region Descriptor n, Word 2 (SMPU_RGD0_WORD2)	32	R/W	0000_0000h	24.3.7/953
40C	Region Descriptor n, Word 3 (SMPU_RGD0_WORD3)	32	R/W	0000_0000h	24.3.8/955
410	Region Descriptor n, Word 0 (SMPU_RGD1_WORD0)	32	R/W	0000_0000h	24.3.5/952
414	Region Descriptor n, Word 1 (SMPU_RGD1_WORD1)	32	R/W	0000_0000h	24.3.6/952

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
418	Region Descriptor n, Word 2 (SMPU_RGD1_WORD2)	32	R/W	0000_0000h	24.3.7/953
41C	Region Descriptor n, Word 3 (SMPU_RGD1_WORD3)	32	R/W	0000_0000h	24.3.8/955
420	Region Descriptor n, Word 0 (SMPU_RGD2_WORD0)	32	R/W	0000_0000h	24.3.5/952
424	Region Descriptor n, Word 1 (SMPU_RGD2_WORD1)	32	R/W	0000_0000h	24.3.6/952
428	Region Descriptor n, Word 2 (SMPU_RGD2_WORD2)	32	R/W	0000_0000h	24.3.7/953
42C	Region Descriptor n, Word 3 (SMPU_RGD2_WORD3)	32	R/W	0000_0000h	24.3.8/955
430	Region Descriptor n, Word 0 (SMPU_RGD3_WORD0)	32	R/W	0000_0000h	24.3.5/952
434	Region Descriptor n, Word 1 (SMPU_RGD3_WORD1)	32	R/W	0000_0000h	24.3.6/952
438	Region Descriptor n, Word 2 (SMPU_RGD3_WORD2)	32	R/W	0000_0000h	24.3.7/953
43C	Region Descriptor n, Word 3 (SMPU_RGD3_WORD3)	32	R/W	0000_0000h	24.3.8/955
440	Region Descriptor n, Word 0 (SMPU_RGD4_WORD0)	32	R/W	0000_0000h	24.3.5/952
444	Region Descriptor n, Word 1 (SMPU_RGD4_WORD1)	32	R/W	0000_0000h	24.3.6/952
448	Region Descriptor n, Word 2 (SMPU_RGD4_WORD2)	32	R/W	0000_0000h	24.3.7/953
44C	Region Descriptor n, Word 3 (SMPU_RGD4_WORD3)	32	R/W	0000_0000h	24.3.8/955
450	Region Descriptor n, Word 0 (SMPU_RGD5_WORD0)	32	R/W	0000_0000h	24.3.5/952
454	Region Descriptor n, Word 1 (SMPU_RGD5_WORD1)	32	R/W	0000_0000h	24.3.6/952
458	Region Descriptor n, Word 2 (SMPU_RGD5_WORD2)	32	R/W	0000_0000h	24.3.7/953
45C	Region Descriptor n, Word 3 (SMPU_RGD5_WORD3)	32	R/W	0000_0000h	24.3.8/955
460	Region Descriptor n, Word 0 (SMPU_RGD6_WORD0)	32	R/W	0000_0000h	24.3.5/952
464	Region Descriptor n, Word 1 (SMPU_RGD6_WORD1)	32	R/W	0000_0000h	24.3.6/952
468	Region Descriptor n, Word 2 (SMPU_RGD6_WORD2)	32	R/W	0000_0000h	24.3.7/953
46C	Region Descriptor n, Word 3 (SMPU_RGD6_WORD3)	32	R/W	0000_0000h	24.3.8/955
470	Region Descriptor n, Word 0 (SMPU_RGD7_WORD0)	32	R/W	0000_0000h	24.3.5/952
474	Region Descriptor n, Word 1 (SMPU_RGD7_WORD1)	32	R/W	0000_0000h	24.3.6/952
478	Region Descriptor n, Word 2 (SMPU_RGD7_WORD2)	32	R/W	0000_0000h	24.3.7/953
47C	Region Descriptor n, Word 3 (SMPU_RGD7_WORD3)	32	R/W	0000_0000h	24.3.8/955
480	Region Descriptor n, Word 0 (SMPU_RGD8_WORD0)	32	R/W	0000_0000h	24.3.5/952
484	Region Descriptor n, Word 1 (SMPU_RGD8_WORD1)	32	R/W	0000_0000h	24.3.6/952
488	Region Descriptor n, Word 2 (SMPU_RGD8_WORD2)	32	R/W	0000_0000h	24.3.7/953
48C	Region Descriptor n, Word 3 (SMPU_RGD8_WORD3)	32	R/W	0000_0000h	24.3.8/955
490	Region Descriptor n, Word 0 (SMPU_RGD9_WORD0)	32	R/W	0000_0000h	24.3.5/952
494	Region Descriptor n, Word 1 (SMPU_RGD9_WORD1)	32	R/W	0000_0000h	24.3.6/952
498	Region Descriptor n, Word 2 (SMPU_RGD9_WORD2)	32	R/W	0000_0000h	24.3.7/953
49C	Region Descriptor n, Word 3 (SMPU_RGD9_WORD3)	32	R/W	0000_0000h	24.3.8/955
4A0	Region Descriptor n, Word 0 (SMPU_RGD10_WORD0)	32	R/W	0000_0000h	24.3.5/952
4A4	Region Descriptor n, Word 1 (SMPU_RGD10_WORD1)	32	R/W	0000_0000h	24.3.6/952
4A8	Region Descriptor n, Word 2 (SMPU_RGD10_WORD2)	32	R/W	0000_0000h	24.3.7/953
4AC	Region Descriptor n, Word 3 (SMPU_RGD10_WORD3)	32	R/W	0000_0000h	24.3.8/955
4B0	Region Descriptor n, Word 0 (SMPU_RGD11_WORD0)	32	R/W	0000_0000h	24.3.5/952

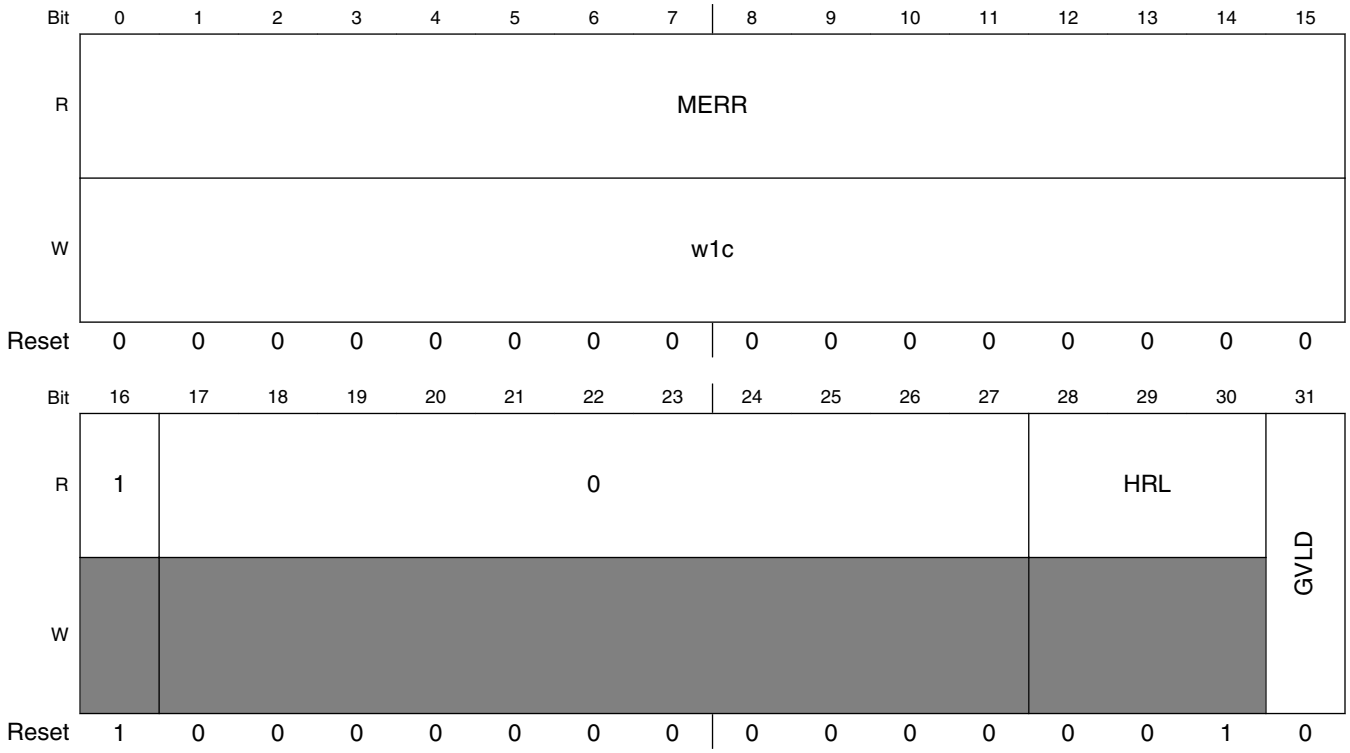
Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4B4	Region Descriptor n, Word 1 (SMPU_RGD11_WORD1)	32	R/W	0000_0000h	24.3.6/952
4B8	Region Descriptor n, Word 2 (SMPU_RGD11_WORD2)	32	R/W	0000_0000h	24.3.7/953
4BC	Region Descriptor n, Word 3 (SMPU_RGD11_WORD3)	32	R/W	0000_0000h	24.3.8/955
4C0	Region Descriptor n, Word 0 (SMPU_RGD12_WORD0)	32	R/W	0000_0000h	24.3.5/952
4C4	Region Descriptor n, Word 1 (SMPU_RGD12_WORD1)	32	R/W	0000_0000h	24.3.6/952
4C8	Region Descriptor n, Word 2 (SMPU_RGD12_WORD2)	32	R/W	0000_0000h	24.3.7/953
4CC	Region Descriptor n, Word 3 (SMPU_RGD12_WORD3)	32	R/W	0000_0000h	24.3.8/955
4D0	Region Descriptor n, Word 0 (SMPU_RGD13_WORD0)	32	R/W	0000_0000h	24.3.5/952
4D4	Region Descriptor n, Word 1 (SMPU_RGD13_WORD1)	32	R/W	0000_0000h	24.3.6/952
4D8	Region Descriptor n, Word 2 (SMPU_RGD13_WORD2)	32	R/W	0000_0000h	24.3.7/953
4DC	Region Descriptor n, Word 3 (SMPU_RGD13_WORD3)	32	R/W	0000_0000h	24.3.8/955
4E0	Region Descriptor n, Word 0 (SMPU_RGD14_WORD0)	32	R/W	0000_0000h	24.3.5/952
4E4	Region Descriptor n, Word 1 (SMPU_RGD14_WORD1)	32	R/W	0000_0000h	24.3.6/952
4E8	Region Descriptor n, Word 2 (SMPU_RGD14_WORD2)	32	R/W	0000_0000h	24.3.7/953
4EC	Region Descriptor n, Word 3 (SMPU_RGD14_WORD3)	32	R/W	0000_0000h	24.3.8/955
4F0	Region Descriptor n, Word 0 (SMPU_RGD15_WORD0)	32	R/W	0000_0000h	24.3.5/952
4F4	Region Descriptor n, Word 1 (SMPU_RGD15_WORD1)	32	R/W	0000_0000h	24.3.6/952
4F8	Region Descriptor n, Word 2 (SMPU_RGD15_WORD2)	32	R/W	0000_0000h	24.3.7/953
4FC	Region Descriptor n, Word 3 (SMPU_RGD15_WORD3)	32	R/W	0000_0000h	24.3.8/955

24.3.1 Control/Error Status Register 0 (SMPU_CESR0)

Address: 0h base + 0h offset = 0h

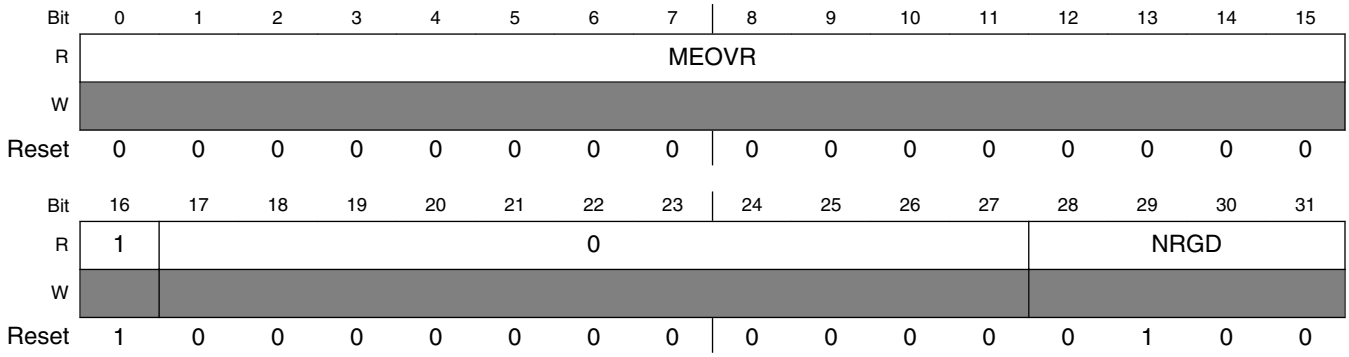


SMPU_CESR0 field descriptions

Field	Description
0–15 MERR	Master n error, where the bus master number matches the bit number Indicates a captured error in EARn and EDRn. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master n. 1 An error has occurred for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

24.3.2 Control/Error Status Register 1 (SMPU_CESR1)

Address: 0h base + 4h offset = 4h



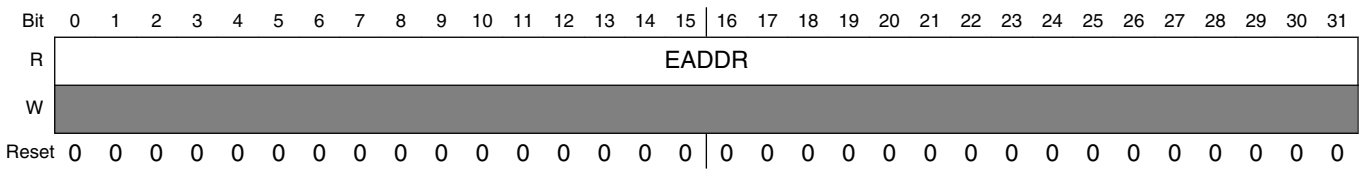
SMPU_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master n error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master n has occurred before the previous error was processed. The details of the first error are recorded in the EARn and EDRn registers and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master n. 1 An error overrun condition has been detected for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. The number of region descriptors is (4 x NRGD). 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

24.3.3 Error Address Register, Bus Master n (SMPU_EAR*n*)

When the SMPU detects an access error on bus master n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] set. Additional information about the faulting access is captured in the corresponding EDRn at the same time.[1]

Address: 0h base + 100h offset + (8d × i), where i=0d to 15d



SMPU_EAR*n* field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error.

1. The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

24.3.4 Error Detail Register, Bus Master n (SMPU_EDR*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time. [2]

Address: 0h base + 104h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

SMPU_EDR*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, the access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference.

Table continues on the next page...

2. The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

SMPU_EDR*n* field descriptions (continued)

Field	Description
	0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference.

24.3.5 Region Descriptor n, Word 0 (SMPU_RGD*n*_WORD0)

The first word of the region descriptor defines the byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 0h base + 400h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

24.3.6 Region Descriptor n, Word 1 (SMPU_RGD*n*_WORD1)

The second word of the region descriptor defines the end address of the memory region. (RGDn_WORD3[VLD]).

Address: 0h base + 404h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. **NOTE:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

24.3.7 Region Descriptor n, Word 2 (SMPU_RGD*n*_WORD2)

RGD_WORD2 defines the access control rights of the memory region. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) operation.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0h base + 408h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD2 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions

Table continues on the next page...

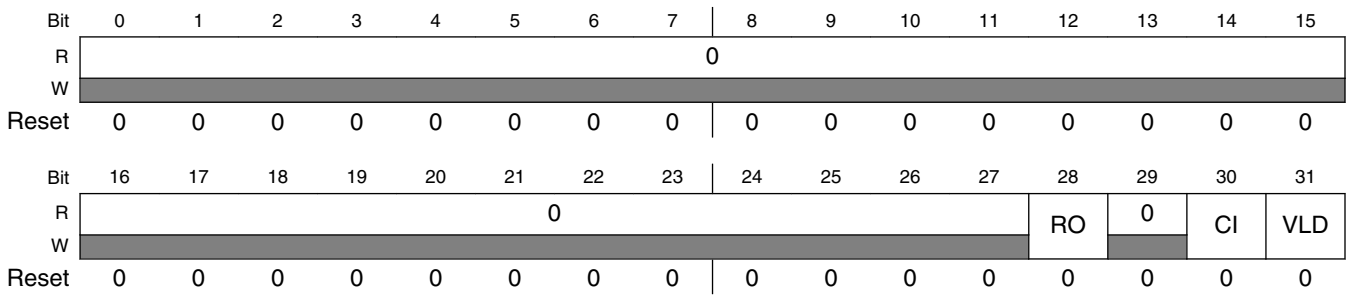
SMPU_RGD*n*_WORD2 field descriptions (continued)

Field	Description
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

24.3.8 Region Descriptor n, Word 3 (SMPU_RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, plus other miscellaneous flags.

Address: 0h base + 40Ch offset + (16d × i), where i=0d to 15d



SMPU_RGD*n*_WORD3 field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **NOTE:** The valid bit of the RGD and the global valid bit have no effect on this bit functionality (once set, even if the RGD is not valid or SMPU is not global enabled, the RO functionality is present). 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **NOTE:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size factoring into the end address. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

24.4 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

24.4.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the connection matrix. The following block diagram represents an individual macro.

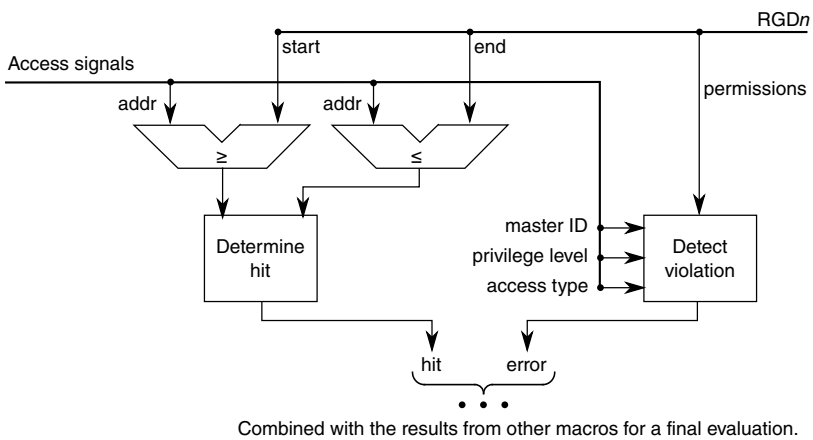


Figure 24-3. SMPU access evaluation macro

Each macro uses the information contained in the access signals (the address, the access type, and the master's ID and privilege level) and the contents of its corresponding region descriptor (RGD*n*) to perform two major functions: Hit determination and Violation determination. The individual hit and error results from each macro are then combined for a final evaluation; see Final evaluation and error terminations.

24.4.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. When RGD*n*_Word3[VLD] is 1, a region hit occurs when the requested address is between the start address (RGD*n*_Word0[SRTADDR]) and the end address (RGD*n*_Word1[ENDADDR]), inclusive.

NOTE

If ENDADDR < SRTADDR, no hit occurs.

NOTE

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

24.4.1.2 Violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master ID, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

Table 24-2. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no execute permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

24.4.2 Final evaluation and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.

3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

NOTE

Similarly, the cache inhibit function gives a higher priority to the less restrictive result of overlapping regions. If two regions are hit and one has cache inhibit set, and the other does not, the result will not include a cache inhibit. It should also be noted that only the hit logic is taken into account for cache inhibit. For example, two region hits, one has protection violation and no cache inhibit, one has no violation and cache inhibit set. The result will be a granted access with no cache inhibit.

24.5 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGD*n*_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD]=0). Next, load the appropriate region descriptors (RGDn), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

Note

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

24.6 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities.

24.6.1 Creating a new memory region

Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

24.6.2 Modifying a region descriptor

Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word 3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

24.6.3 Removing a region descriptor

Clearing RGD*n*_Word3[VLD] deletes an existing region descriptor.

24.6.4 Accessing the SMPU

Allocate a region descriptor to restrict SMPU access to supervisor mode from specific masters as appropriate.

24.6.5 Detecting an access error

The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals which error registers contain captured fault data. Once the error information is retrieved, write a 1 to the appropriate CESR0[MERR] bit to clear it, thereby rearming the error capture logic associated with the EAR*n* and EDR*n* registers.

24.6.6 Overlapping region descriptors

Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

24.6.7 EAR*n* address capture considerations

When the SMPU detects an access error and the cache is enabled, the captured address in EAR*n* can differ from the expected address due to pipeline execution.

An instruction fetch for a master can cause the master's corresponding bit in CESR1[MEOVR] to set to 1 automatically in certain circumstances. For example, when the pipeline mechanism prefetches several instructions from multiple addresses in the same memory area, more than one fetch can result in an access error for the master.

Chapter 25
Semaphores2 (SEMA42)

25.1 Chip-specific SEMA42 information

The SEMA42 module uses the same logical bus master assignments as the SMPUs. The logical bus master assignments for the SEMA42 are same as SMPU, see SMPU_x Logical bus Master Assignments.

25.2 Introduction

The SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

25.2.1 Features

- Supports 16 hardware-enforced gates in a multi-processor configuration
  - Gates appear as an 16-entry byte-size array with read and write accesses.
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.

- 16-state implementation

  if gate = 0x0, then state = unlocked
  if gate = 0x1, then state = locked by processor (master) 0
  if gate = 0x2, then state = locked by processor (master) 1
  …
  if gate = 0xF, then state = locked by processor (master) 14
- Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
- Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.

- Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2), gate (n-1) include the finite state machines implementing the semaphore gates.

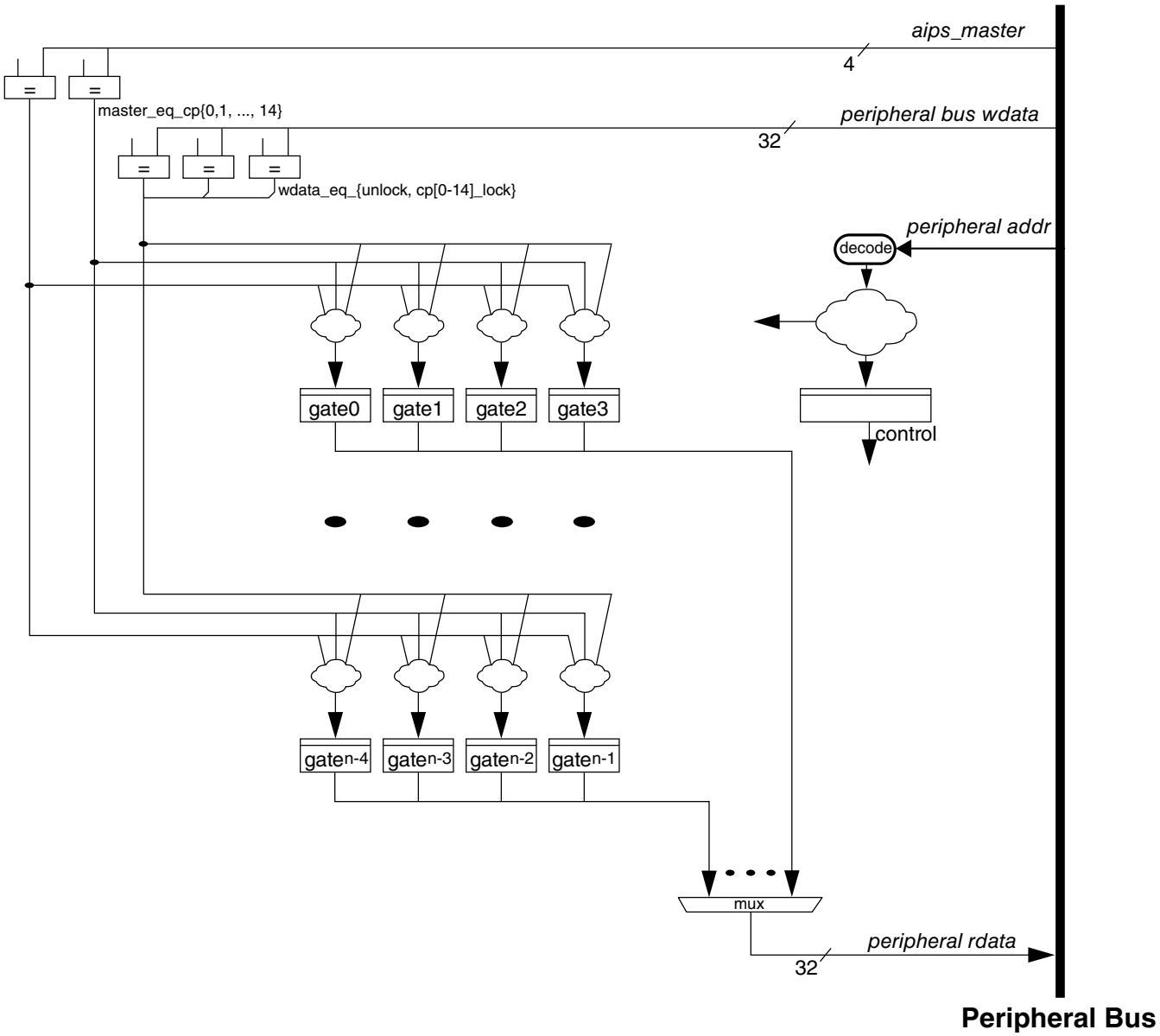


Figure 25-1. Semaphores2 block diagram

25.3 Memory map/register definition

Only Supervisor Mode accesses are allowed on these registers. User accesses will be terminated with an error.

SEMA42 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Gate Register (SEMA42_GATE0)	8	R/W	00h	25.3.1/964

Table continues on the next page...

SEMA42 memory map (continued)

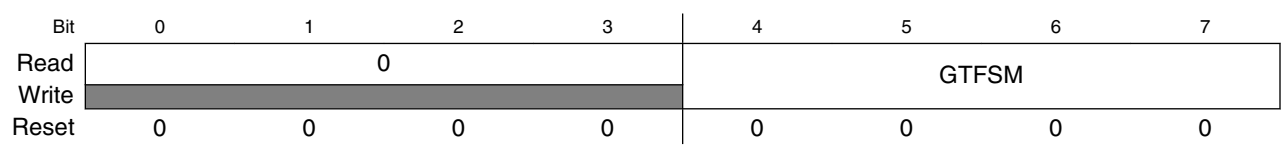
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1	Gate Register (SEMA42_GATE1)	8	R/W	00h	25.3.1/964
2	Gate Register (SEMA42_GATE2)	8	R/W	00h	25.3.1/964
3	Gate Register (SEMA42_GATE3)	8	R/W	00h	25.3.1/964
4	Gate Register (SEMA42_GATE4)	8	R/W	00h	25.3.1/964
5	Gate Register (SEMA42_GATE5)	8	R/W	00h	25.3.1/964
6	Gate Register (SEMA42_GATE6)	8	R/W	00h	25.3.1/964
7	Gate Register (SEMA42_GATE7)	8	R/W	00h	25.3.1/964
8	Gate Register (SEMA42_GATE8)	8	R/W	00h	25.3.1/964
9	Gate Register (SEMA42_GATE9)	8	R/W	00h	25.3.1/964
A	Gate Register (SEMA42_GATE10)	8	R/W	00h	25.3.1/964
B	Gate Register (SEMA42_GATE11)	8	R/W	00h	25.3.1/964
C	Gate Register (SEMA42_GATE12)	8	R/W	00h	25.3.1/964
D	Gate Register (SEMA42_GATE13)	8	R/W	00h	25.3.1/964
E	Gate Register (SEMA42_GATE14)	8	R/W	00h	25.3.1/964
F	Gate Register (SEMA42_GATE15)	8	R/W	00h	25.3.1/964
40	Reset Gate Write (SEMA42_RSTGT_W)	16	W	See section	25.3.2/965
40	Reset Gate Read (SEMA42_RSTGT_R)	16	R	0000h	25.3.3/967

25.3.1 Gate Register (SEMA42_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d



SEMA42_GATE*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 GTFSM	Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14.

25.3.2 Reset Gate Write (SEMA42_RSTGT_W)

NOTE

This section and the following section, Reset Gate Read (SEMA42_RSTGT_R), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is written. Reset Gate Read (SEMA42_RSTGT_R) shows how the register fields appear when the register is read.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that

required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same domain to force the clearing of the specified gate(s). The required access pattern is:

1. A domain performs a 16-bit write to the SEMA42_RSTGT memory location. The most significant byte (SEMA42_RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the SEMA42_RSTGT location. For this write, the upper byte (SEMA42_RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (SEMA42_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.
3. Reads of the SEMA42_RSTGT location return information on the 2-bit state machine (SEMA42_RSTGT[RSTGSM]) that implements this function, the domain performing the reset (SEMA42_RSTGT[RSTGMS]), and the gate number(s) last cleared (SEMA42_RSTGT[RSTGTN]). Reads of the SEMA42_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read																
Write	RSTGDP								RSTGTN							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is not applicable for writes.

SEMA42_RSTGT_W field descriptions

Field	Description
0–7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

25.3.3 Reset Gate Read (SEMA42_RSTGT_R)

This section describes how the Reset Gate register fields appear when the register is read. See Reset Gate Write (SEMA42_RSTGT_W) for the description of the register's overall operation and how the register fields appear when the register is written.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ROZ		RSTGSM		RSTGMS				RSTGTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA42_RSTGT_R field descriptions

Field	Description
0–1 ROZ	This field always returns the value 0 when read.
2–3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA42_RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state. 11 This state encoding is never used and therefore reserved.
4–7 RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

25.4 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in Figure 25-2.

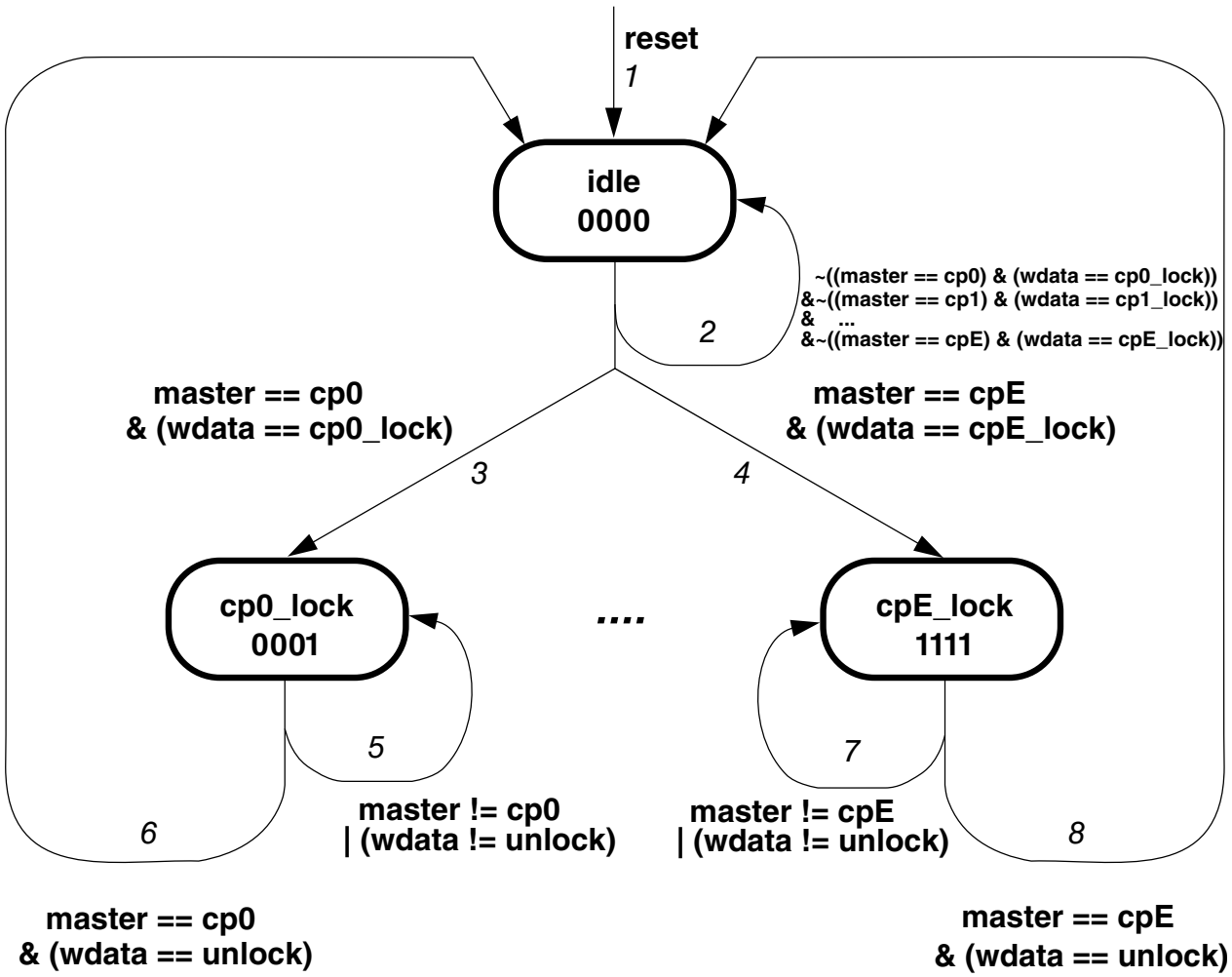


Figure 25-2. SEMA42_GATEn state machine

The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the Semaphores module.

The state transitions for SEMA42_GATEn are defined in Table 25-1.

Table 25-1. SEMA42_GATEn state transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.

Table continues on the next page...

Table 25-1. SEMA42_GATEn state transitions (continued)

Current State	Next State	Transition	Description
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

The following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

Chapter 26
Development Trigger Semaphore (DTS)

26.1 Chip-specific DTS information

Writes from Nexus to DTS are little endian. For example, writing 0x00000001 leads to 0x00000001 being reflected in DTS registers.

26.2 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2012 Nexus standards, which use the Nexus Auxiliary port.

26.3 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output (DTO). The registers are as follows:

- DTS_SEMAPHORE register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to

detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.

- DTS_SEMAPHORE_B register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_STARTUP register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in the following figure.

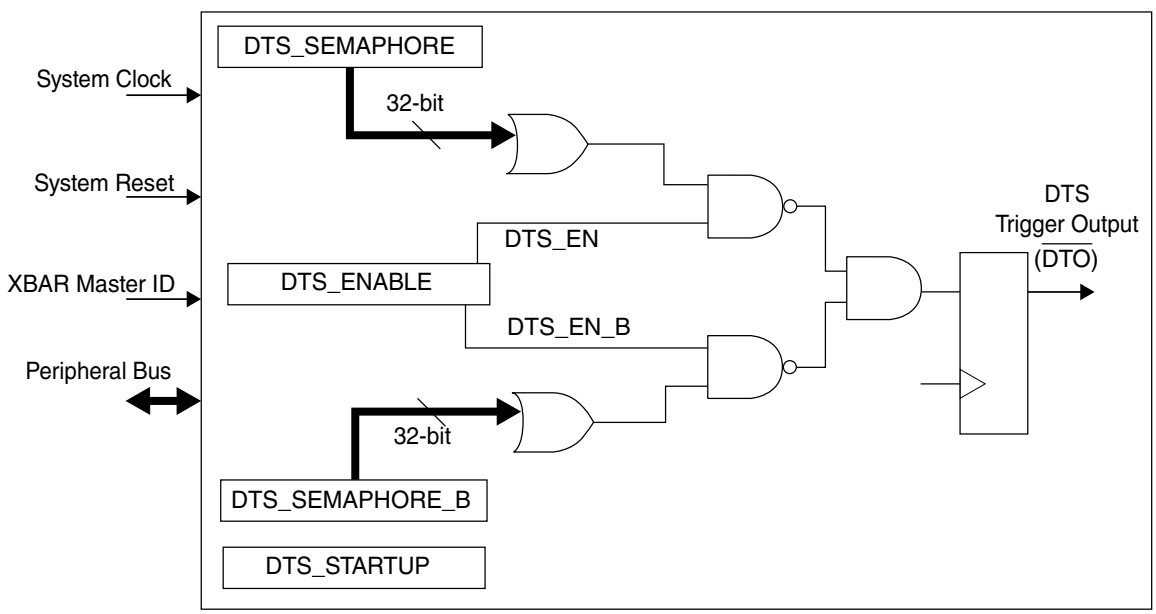


Figure 26-1. DTS block diagram

The DTS Trigger Output (DTO) signal is connected to one of the EVTO inputs of the Nexus Port Controller (NPC). The other EVTO inputs are connected to the other Nexus modules in the device. DTO is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note

When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1 or DTS_ENABLE[DTS_EN_B] = 0b1), all other Nexus EVTO functions associated should be disabled by the tool and EVTO becomes the DTO. Unlike the EVTO function that only asserts for one clock, the DTO function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

The following figure shows an example of the chain of events that begins with setting of a bit in the DTS_SEMPAHORE register and the clearing of the register caused by a Nexus read.

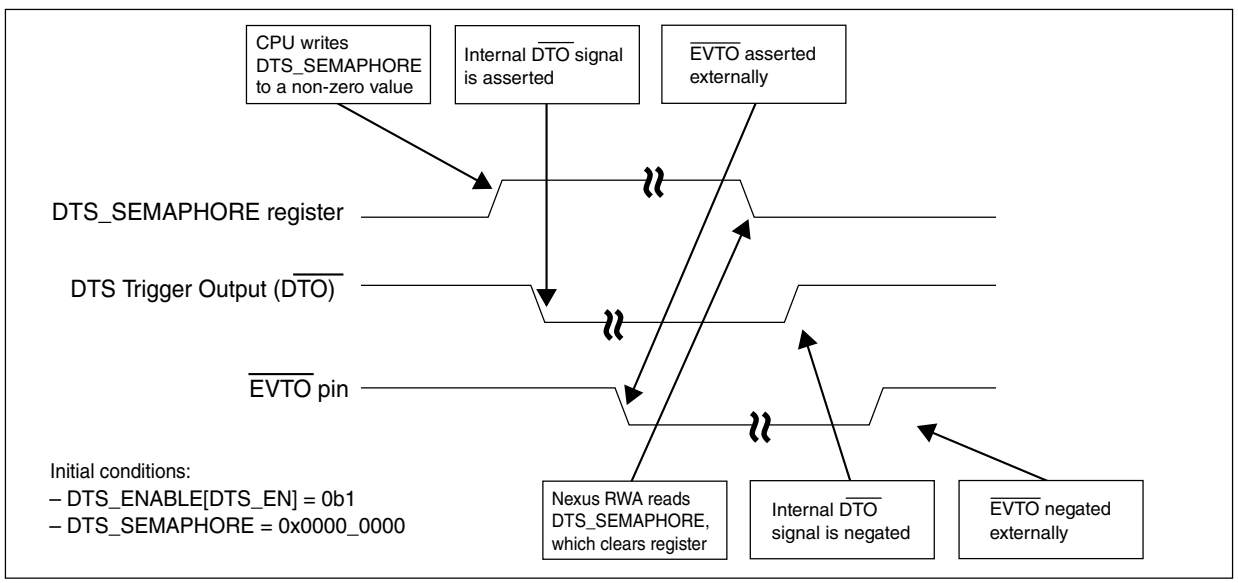


Figure 26-2. DTO event sequence example

26.4 DTS device connections

The following figure shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

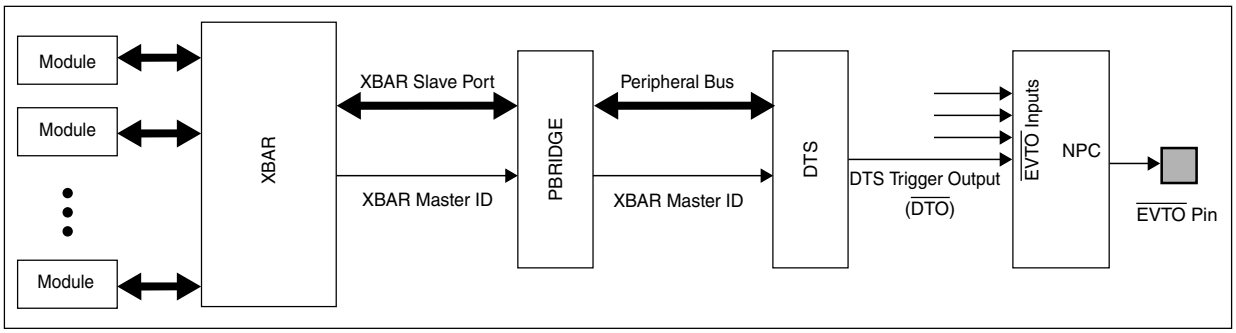


Figure 26-3. DTS device connections

The registers have limited access as described in DTS register access. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access.

Note

Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

26.4.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in the following table. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 26-1. DTS register access effects

Register	32-bit read				32-bit write			
	RWA[1]	Core	eDMA	Other crossbar masters	RWA [1]	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear[2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear[2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note

The XBAR Master ID should not be confused with the Master Port number of the XBAR. See the Crossbar Switch (XBAR) chapter for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

26.5 Memory mapped registers

Only certain types of accesses are allowed to the DTS registers. See the description for each register to determine the accesses that are allowed.

DTS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Output Enable Register (DTS_ENABLE)	32	R/W	0000_0000h	26.5.1/975
4	Startup Register (DTS_STARTUP)	32	R/W	0000_0000h	26.5.2/977
8	Semaphore Register (DTS_SEMAPHORE)	32	R/W	FFFF_FFFFh	26.5.3/977
C	Semaphore Extension (DTS_SEMAPHORE_B)	32	R/W	FFFF_FFFFh	26.5.4/978

26.5.1 Output Enable Register (DTS_ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

NOTE

Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved														DTS_EN_B	DTS_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DTS_ENABLE field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 DTS_EN_B	DTS Enable B. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
31 DTS_EN	DTS Enable. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

26.5.2 Startup Register (DTS_STARTUP)

The DTS_STARTUP register is used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA).

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DTS_STARTUP field descriptions

Field	Description
0–31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

26.5.3 Semaphore Register (DTS_SEMAPHORE)

The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE field descriptions

Field	Description
0–31 ST	Semaphore Trigger. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset.

DTS_SEMAPHORE field descriptions (continued)

Field	Description
	• A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

26.5.4 Semaphore Extension (DTS_SEMAPHORE_B)

The DTS_SEMAPHORE_B register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_B																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE_B field descriptions

Field	Description
0–31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset. • A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

26.6 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note

It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

The following table shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE

register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

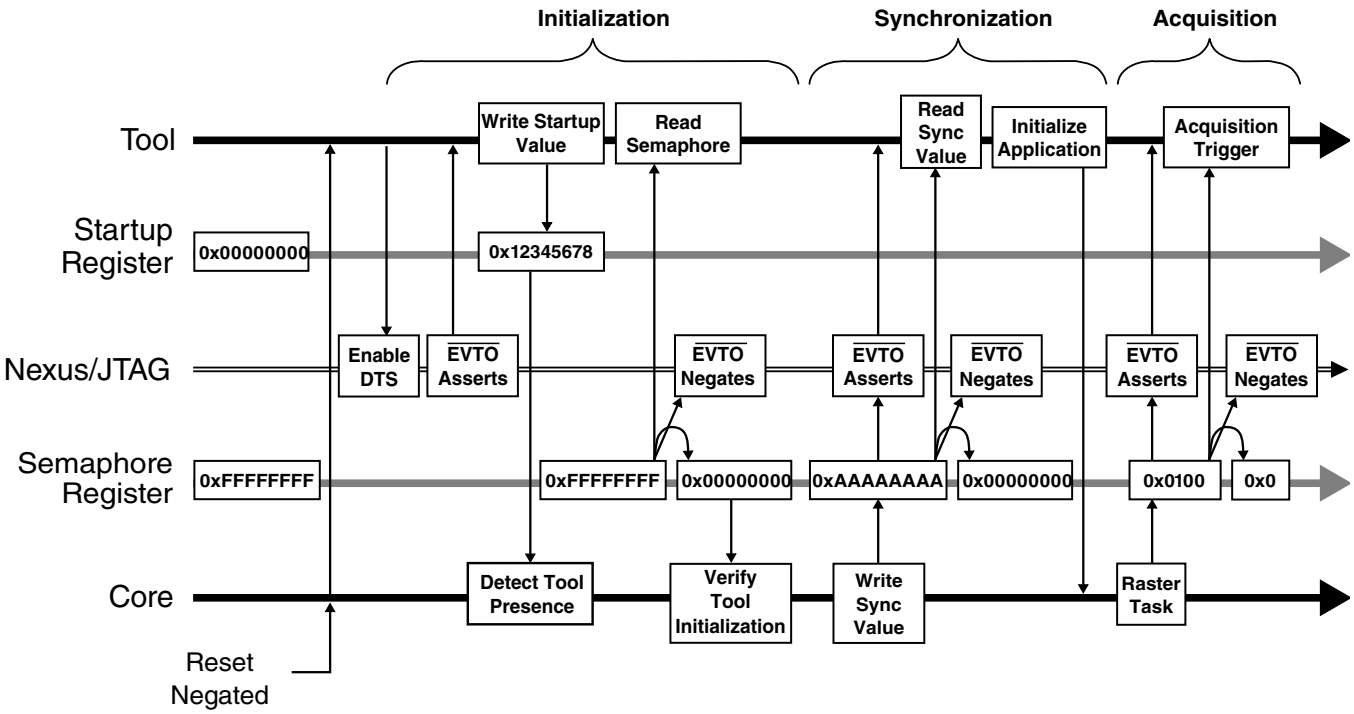


Figure 26-4. DTS startup sequence example

Chapter 27
Wakeup Unit (WKPU)

27.1 Chip-specific WKPU information

27.1.1 Chip-specific WKPU configuration

The S32R274 device WKPU supports only one external source from the NMI pad that can cause non-maskable interrupts to on-chip cores, and wakeup events to the system.

For Pad details, see "IO Signal Description and Input multiplexing tables" spreadsheet attached with this document.

27.1.2 Register reset values

Some of the registers in this module have chip-specific reset values. They are listed below.

Register	Reset value
WKPU_NCR	0x0000_0000

27.1.3 Clocking limitation for Core1 and Core2 NMIs

For the NMI interface of Core1 and Core2 to work, the CORE0_CLK should be kept on.

27.2 Introduction

The Wakeup Unit (WKPU) supports three external sources that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. The figure below shows the WKPU block diagram and its interfaces to other system components.

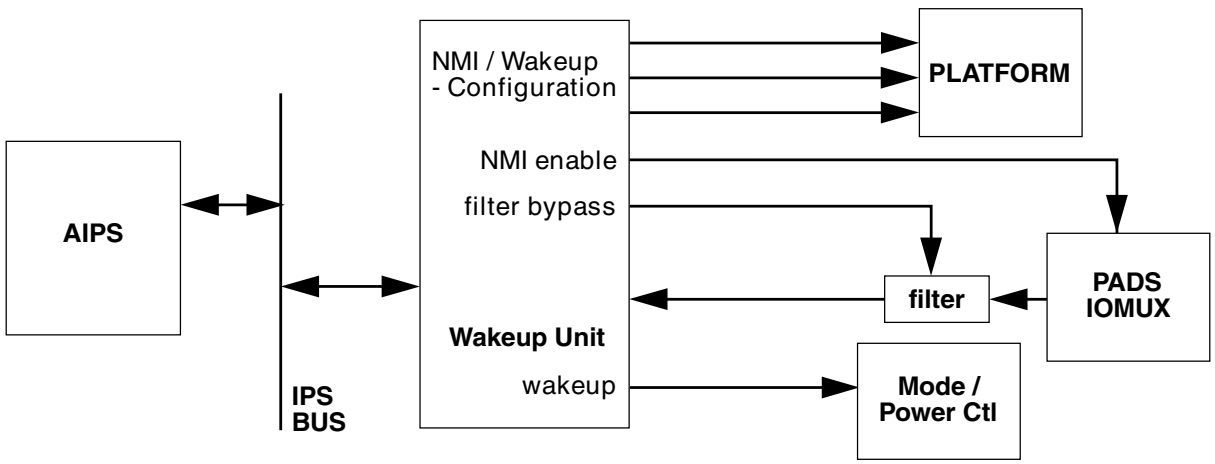


Figure 27-1. WKPU and connected system components

27.3 Features

The WKPU supports these features:

- Non-maskable Interrupt support with:
  - External NMI sources
  - Analog glitch filters
- Independent interrupt destination for each core:
  - Non-maskable interrupt
- Active edge selection control for events to each core
- Configurable system wakeup triggering from NMI source(s)

27.4 External signal description

The WKPU has signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

27.5 WKPU memory map and register definition

This section provides a detailed description of all registers accessible in the WKPU module.

NOTE

Reserved registers will read as 0, writes will have no effect. Transfer error will be generated when trying to access completely reserved register space.

WKPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	NMI Status Flag Register (WKPU_NSR)	32	w1c	0000_0000h	27.5.1/983
8	NMI Configuration Register (WKPU_NCR)	32	R/W	See section	27.5.2/985

27.5.1 NMI Status Flag Register (WKPU_NSR)

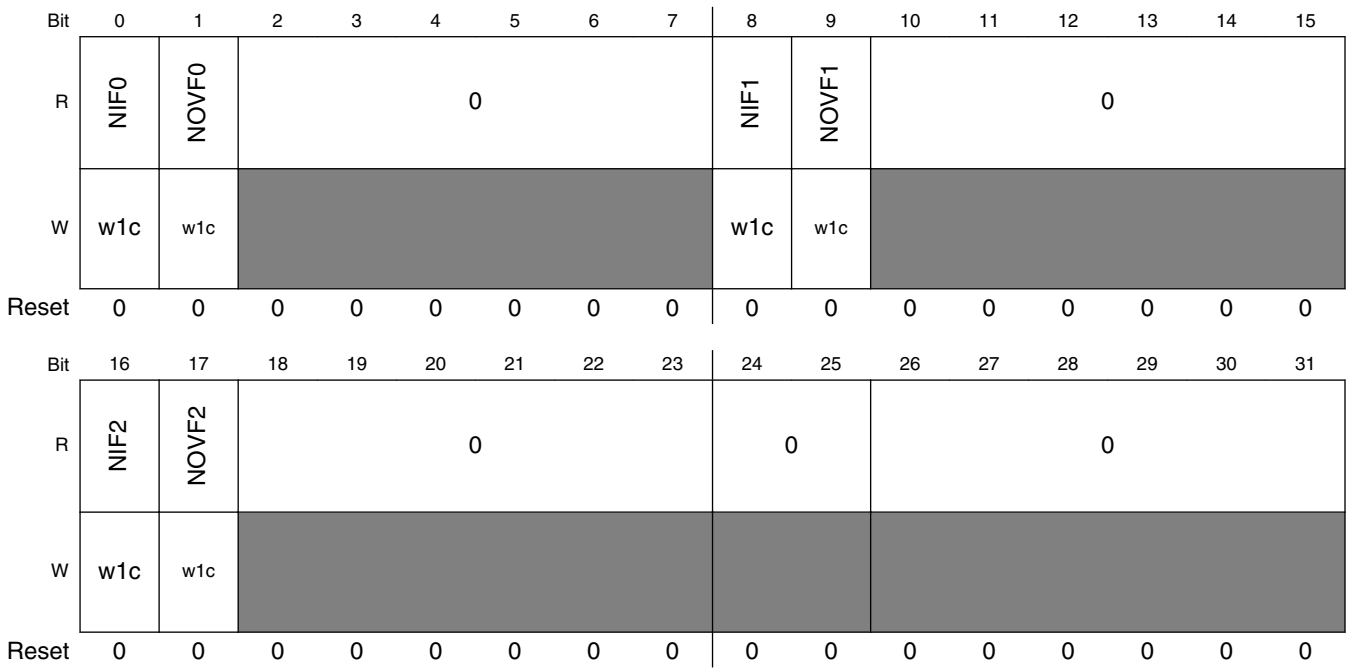
This register holds the non-maskable interrupt status flags.

NOTE

This register is accessible by 8-, 16-, and 32-bit read/write operations.

Access: User read/write

Address: 0h base + 0h offset = 0h



WKPU_NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE0 and NFEE0 has occurred
1 NOVF0	NMI Overrun Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 0 No overrun has occurred on NMI input 0 1 An overrun has occurred on NMI input 0
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 NIF1	NMI Status Flag 1. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE1 or NFEE1 set), NIF1 causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE1 and NFEE1 has occurred
9 NOVF1	NMI Overrun Status Flag 1. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF1 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE1 or NFEE1 set), NOVF1 causes an interrupt request. 0 No overrun has occurred on NMI input 1 1 An overrun has occurred on NMI input 1
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

WKPU_NSR field descriptions (continued)

Field	Description
16 NIF2	NMI Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE2 or NFEE2 set), NIF2 causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE2 and NFEE2 has occurred
17 NOVF2	NMI Overrun Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF2 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE2 or NFEE2 set), NOVF2 causes an interrupt request. 0 No overrun has occurred on NMI input 2 1 An overrun has occurred on NMI input 2
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.5.2 NMI Configuration Register (WKPU_NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

NOTE

- This register is accessible by 8-, 16-, and 32-bit read/write operations.
- Writing a 0 to both NREE[*n*] and NFEE[*n*] disables the NMI functionality completely (i.e., no system wakeup or interrupt will be generated on any pad activity).

Access: User read/write

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	NLOCK1	NDSS1		NWRE1	0	NREE1	NFEE1	NFE1
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	NFE2	Reserved							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the chip-specific WKPU information for the reset value of this register.

WKPU_NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.
1–2 NDSS0	NMI Destination Source Select 0. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 No NMI, critical Interrupt, or machine check request generated.
3 NWRE0	NMI Wakeup Request Enable 0 **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NIF0 bit are disabled. 1 A set NIF0 bit or set NOVF0 bit causes a system wakeup request.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 NREE0	NMI Rising-edge Events Enable 0. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 NFEE0	NMI Falling-edge Events Enable 0. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 NFE0	NMI Filter Enable 0. Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
8 NLOCK1	NMI Configuration Lock Register 1. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
9–10 NDSS1	NMI Destination Source Select 1. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 No NMI, critical Interrupt, or machine check request generated.
11 NWRE1	NMI Wakeup Request Enable 1. **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NIF1 bit are disabled 1 A set NIF1 bit or set NOVF1 bit causes a system wakeup request
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 NREE1	NMI Rising-edge Events Enable 1. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 NFEE1	NMI Falling-edge Events Enable 1. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 NFE1	NMI Filter Enable 1. Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
16 NLOCK2	NMI Configuration Lock Register 2. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.
17–18 NDSS2	NMI Destination Source Select 2. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 No NMI, critical Interrupt, or machine check request generated.
19 NWRE2	NMI Wakeup Request Enable 2. **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check.

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
	0 System wakeup requests from the corresponding NIF2 bit are disabled 1 A set NIF2 bit or set NOVF2 bit causes a system wakeup request
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 NREE2	NMI Rising-edge Events Enable 2. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 NFEE2	NMI Falling-edge Events Enable 2. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 NFE2	NMI Filter Enable 2. Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
24–31 Reserved	This field is reserved. Always write 0 to this field.

27.6 Functional description

This section provides a functional description of the WKPU.

27.6.1 Non-maskable interrupts

The WKPU supports the generation of 3 types of interrupts per NMI input to the SoC. The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event, though it creates an overrun condition.

Each NMI passes through a bypassable analog glitch filter.

NOTE

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

NOTE

The figure below represents a generic configuration and might not represent this particular device's configuration. See the chip-specific information for details on this chip's WKPU.

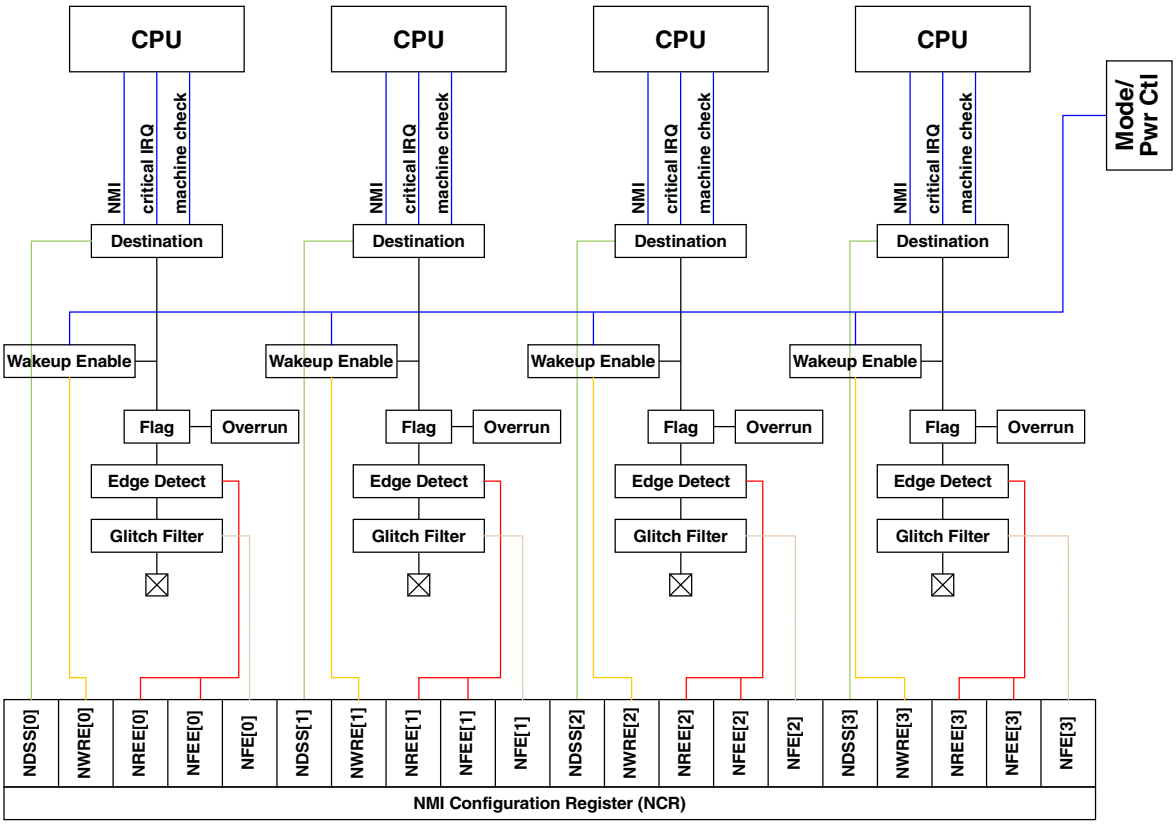


Figure 27-2. NMI pad diagram

27.6.1.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see NMI Configuration Register (WKPU_NCR)). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note

After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI. Please see chip specific section for details on NMI implementation.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See NMI Configuration Register (WKPU_NCR) for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see NMI Status Flag Register (WKPU_NSR)). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

Note

The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

During an NMI ISR, on wakeup of the SoC from an NMI, any writes to ECC-protected memory must have the correct ECC.

Chapter 28
Interrupt Controller (INTC)

28.1 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 32 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

28.2 Block diagram

The following figure shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

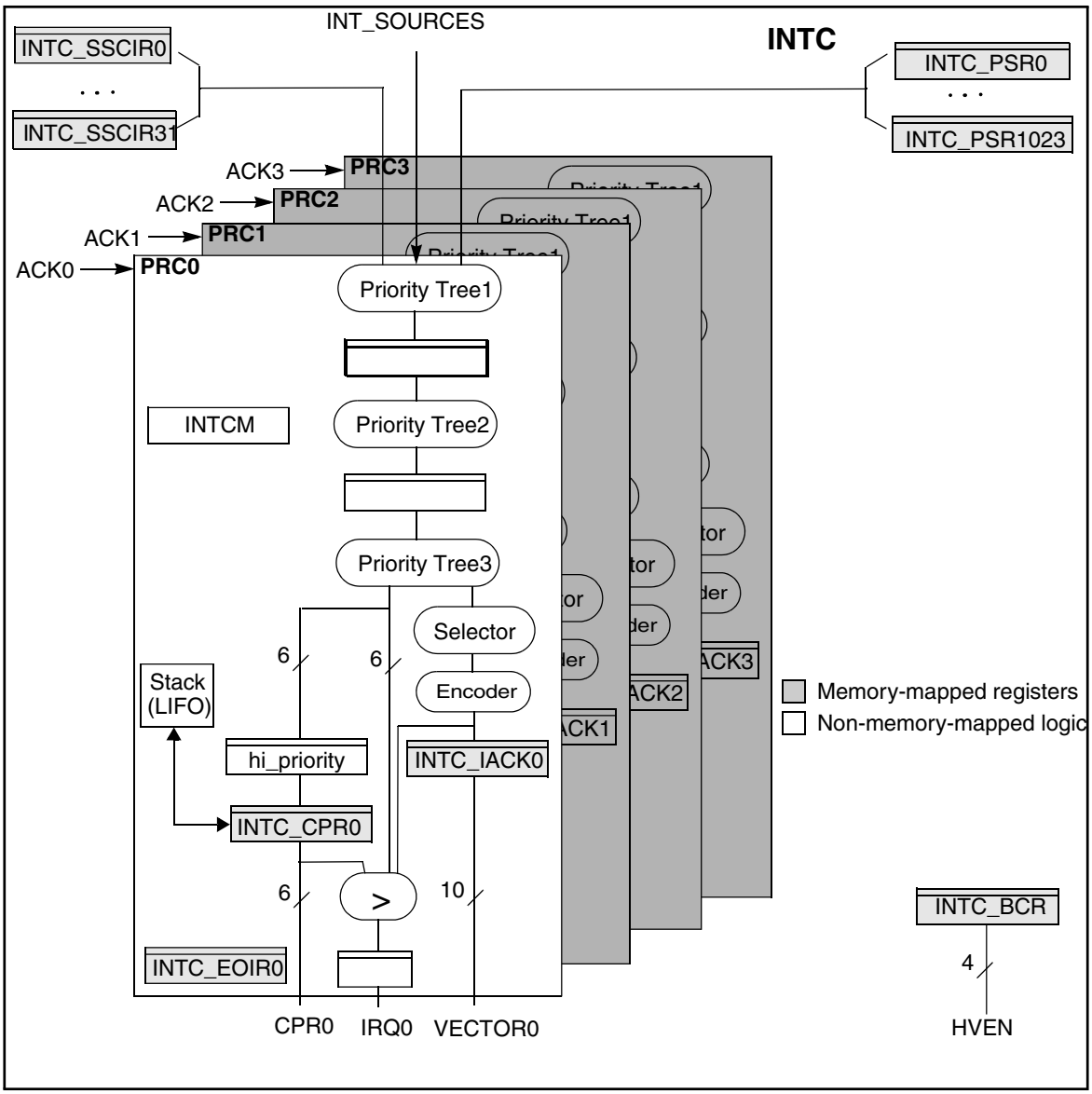


Figure 28-1. Block diagram for an INTC with four processors

28.3 Features

- Each peripheral interrupt source is software-steerable to any combination of interrupt request outputs to the following:
  - Processor 0
  - Processor 1
  - Processor 2
- 16 software-settable interrupt request sources

- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 32 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request
- Monitor
  - The Interrupt Controller Monitor (INTCM) provides a way to set a maximum timer count for the interrupt latency from interrupt request (IRQ) to interrupt acknowledge (IACK) or from IRQ to return from interrupt (RFI). The purpose of this function is to provide a mechanism to monitor the latency of interrupt sources for a corresponding Processor to ensure that these critical interrupts execute within the expected window of time, increasing the reliability of the device. The hardware for this function is isolated in its own hierarchy to decrease the risk of fault interference.

28.4 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point

for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

28.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN_PRCn bit in the INTC_BCR is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers. Reading the INTC_IACKR*n* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

28.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVENn bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKRn, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the

associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

28.5 Memory map and register definition

A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

With the exception of the INTC_SSCIR *n* and INTC_PSR *n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

When writing to INTC_PSRs or INTC_SSCIRs, the following restrictions (illustrated in Figure 28-2) apply:

- For PSRs, write accesses must not span 16-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).
- For SSCIRs, write accesses must not span 8-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).

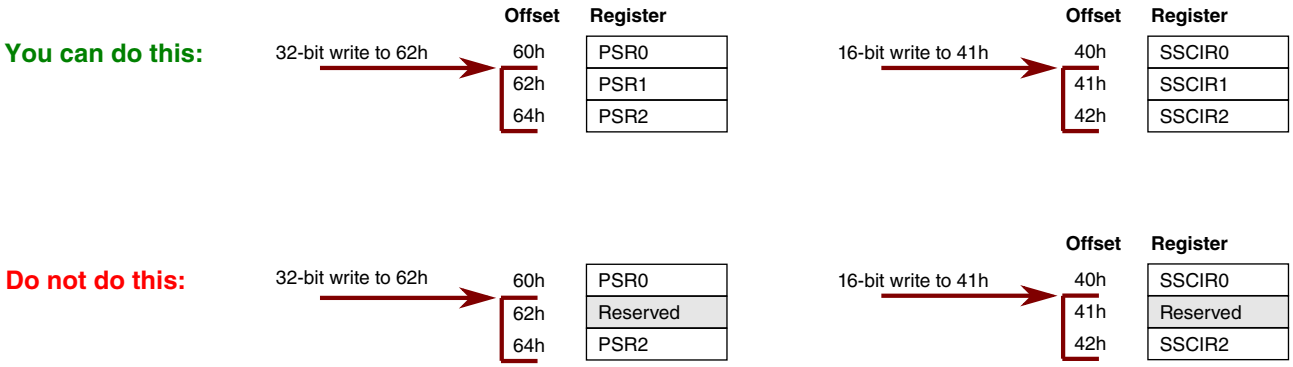


Figure 28-2. INTC_PSR*n* and INTC_SSCIR*n* write restrictions

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR *n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR *n*) does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Block Configuration Register (INTC_BCR)	32	R/W	0000_0000h	28.5.1/1046
10	INTC Current Priority Register for Processor 0 (INTC_CPR0)	32	R/W	0000_001Fh	28.5.2/1047
14	INTC Current Priority Register for Processor 1 (INTC_CPR1)	32	R/W	0000_001Fh	28.5.3/1048
18	INTC Current Priority Register for Processor 2 (INTC_CPR2)	32	R/W	0000_001Fh	28.5.4/1049
20	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	32	R/W	0000_0000h	28.5.5/1050
24	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	32	R/W	0000_0000h	28.5.6/1051
28	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	32	R/W	0000_0000h	28.5.7/1051
30	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	32	W	0000_0000h	28.5.8/1052
34	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	32	W	0000_0000h	28.5.9/1053
38	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	32	W	0000_0000h	28.5.10/ 1053
40	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	28.5.11/ 1054
41	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	28.5.11/ 1054
42	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	28.5.11/ 1054
43	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	28.5.11/ 1054
44	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	28.5.11/ 1054

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
45	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	28.5.11/ 1054
46	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	28.5.11/ 1054
47	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	28.5.11/ 1054
48	INTC Software Set/Clear Interrupt Register (INTC_SSCIR8)	8	R/W	00h	28.5.11/ 1054
49	INTC Software Set/Clear Interrupt Register (INTC_SSCIR9)	8	R/W	00h	28.5.11/ 1054
4A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR10)	8	R/W	00h	28.5.11/ 1054
4B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR11)	8	R/W	00h	28.5.11/ 1054
4C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR12)	8	R/W	00h	28.5.11/ 1054
4D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR13)	8	R/W	00h	28.5.11/ 1054
4E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR14)	8	R/W	00h	28.5.11/ 1054
4F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR15)	8	R/W	00h	28.5.11/ 1054
60	INTC Priority Select Register (INTC_PSR0)	16	R/W	8000h	28.5.12/ 1055
62	INTC Priority Select Register (INTC_PSR1)	16	R/W	8000h	28.5.12/ 1055
64	INTC Priority Select Register (INTC_PSR2)	16	R/W	8000h	28.5.12/ 1055
66	INTC Priority Select Register (INTC_PSR3)	16	R/W	8000h	28.5.12/ 1055
68	INTC Priority Select Register (INTC_PSR4)	16	R/W	8000h	28.5.12/ 1055
6A	INTC Priority Select Register (INTC_PSR5)	16	R/W	8000h	28.5.12/ 1055
6C	INTC Priority Select Register (INTC_PSR6)	16	R/W	8000h	28.5.12/ 1055
6E	INTC Priority Select Register (INTC_PSR7)	16	R/W	8000h	28.5.12/ 1055
70	INTC Priority Select Register (INTC_PSR8)	16	R/W	8000h	28.5.12/ 1055
72	INTC Priority Select Register (INTC_PSR9)	16	R/W	8000h	28.5.12/ 1055
74	INTC Priority Select Register (INTC_PSR10)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
76	INTC Priority Select Register (INTC_PSR11)	16	R/W	8000h	28.5.12/ 1055
78	INTC Priority Select Register (INTC_PSR12)	16	R/W	8000h	28.5.12/ 1055
7A	INTC Priority Select Register (INTC_PSR13)	16	R/W	8000h	28.5.12/ 1055
7C	INTC Priority Select Register (INTC_PSR14)	16	R/W	8000h	28.5.12/ 1055
7E	INTC Priority Select Register (INTC_PSR15)	16	R/W	8000h	28.5.12/ 1055
80	INTC Priority Select Register (INTC_PSR16)	16	R/W	8000h	28.5.12/ 1055
82	INTC Priority Select Register (INTC_PSR17)	16	R/W	8000h	28.5.12/ 1055
84	INTC Priority Select Register (INTC_PSR18)	16	R/W	8000h	28.5.12/ 1055
86	INTC Priority Select Register (INTC_PSR19)	16	R/W	8000h	28.5.12/ 1055
88	INTC Priority Select Register (INTC_PSR20)	16	R/W	8000h	28.5.12/ 1055
8A	INTC Priority Select Register (INTC_PSR21)	16	R/W	8000h	28.5.12/ 1055
8C	INTC Priority Select Register (INTC_PSR22)	16	R/W	8000h	28.5.12/ 1055
8E	INTC Priority Select Register (INTC_PSR23)	16	R/W	8000h	28.5.12/ 1055
90	INTC Priority Select Register (INTC_PSR24)	16	R/W	8000h	28.5.12/ 1055
92	INTC Priority Select Register (INTC_PSR25)	16	R/W	8000h	28.5.12/ 1055
94	INTC Priority Select Register (INTC_PSR26)	16	R/W	8000h	28.5.12/ 1055
96	INTC Priority Select Register (INTC_PSR27)	16	R/W	8000h	28.5.12/ 1055
98	INTC Priority Select Register (INTC_PSR28)	16	R/W	8000h	28.5.12/ 1055
9A	INTC Priority Select Register (INTC_PSR29)	16	R/W	8000h	28.5.12/ 1055
9C	INTC Priority Select Register (INTC_PSR30)	16	R/W	8000h	28.5.12/ 1055
9E	INTC Priority Select Register (INTC_PSR31)	16	R/W	8000h	28.5.12/ 1055
A0	INTC Priority Select Register (INTC_PSR32)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A2	INTC Priority Select Register (INTC_PSR33)	16	R/W	8000h	28.5.12/ 1055
A4	INTC Priority Select Register (INTC_PSR34)	16	R/W	8000h	28.5.12/ 1055
A6	INTC Priority Select Register (INTC_PSR35)	16	R/W	8000h	28.5.12/ 1055
A8	INTC Priority Select Register (INTC_PSR36)	16	R/W	8000h	28.5.12/ 1055
AA	INTC Priority Select Register (INTC_PSR37)	16	R/W	8000h	28.5.12/ 1055
AC	INTC Priority Select Register (INTC_PSR38)	16	R/W	8000h	28.5.12/ 1055
AE	INTC Priority Select Register (INTC_PSR39)	16	R/W	8000h	28.5.12/ 1055
B0	INTC Priority Select Register (INTC_PSR40)	16	R/W	8000h	28.5.12/ 1055
B2	INTC Priority Select Register (INTC_PSR41)	16	R/W	8000h	28.5.12/ 1055
B4	INTC Priority Select Register (INTC_PSR42)	16	R/W	8000h	28.5.12/ 1055
B6	INTC Priority Select Register (INTC_PSR43)	16	R/W	8000h	28.5.12/ 1055
B8	INTC Priority Select Register (INTC_PSR44)	16	R/W	8000h	28.5.12/ 1055
BA	INTC Priority Select Register (INTC_PSR45)	16	R/W	8000h	28.5.12/ 1055
BC	INTC Priority Select Register (INTC_PSR46)	16	R/W	8000h	28.5.12/ 1055
BE	INTC Priority Select Register (INTC_PSR47)	16	R/W	8000h	28.5.12/ 1055
C0	INTC Priority Select Register (INTC_PSR48)	16	R/W	8000h	28.5.12/ 1055
C2	INTC Priority Select Register (INTC_PSR49)	16	R/W	8000h	28.5.12/ 1055
C4	INTC Priority Select Register (INTC_PSR50)	16	R/W	8000h	28.5.12/ 1055
C6	INTC Priority Select Register (INTC_PSR51)	16	R/W	8000h	28.5.12/ 1055
C8	INTC Priority Select Register (INTC_PSR52)	16	R/W	8000h	28.5.12/ 1055
CA	INTC Priority Select Register (INTC_PSR53)	16	R/W	8000h	28.5.12/ 1055
CC	INTC Priority Select Register (INTC_PSR54)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CE	INTC Priority Select Register (INTC_PSR55)	16	R/W	8000h	28.5.12/ 1055
D0	INTC Priority Select Register (INTC_PSR56)	16	R/W	8000h	28.5.12/ 1055
D2	INTC Priority Select Register (INTC_PSR57)	16	R/W	8000h	28.5.12/ 1055
D4	INTC Priority Select Register (INTC_PSR58)	16	R/W	8000h	28.5.12/ 1055
D6	INTC Priority Select Register (INTC_PSR59)	16	R/W	8000h	28.5.12/ 1055
D8	INTC Priority Select Register (INTC_PSR60)	16	R/W	8000h	28.5.12/ 1055
DA	INTC Priority Select Register (INTC_PSR61)	16	R/W	8000h	28.5.12/ 1055
DC	INTC Priority Select Register (INTC_PSR62)	16	R/W	8000h	28.5.12/ 1055
DE	INTC Priority Select Register (INTC_PSR63)	16	R/W	8000h	28.5.12/ 1055
E0	INTC Priority Select Register (INTC_PSR64)	16	R/W	8000h	28.5.12/ 1055
E2	INTC Priority Select Register (INTC_PSR65)	16	R/W	8000h	28.5.12/ 1055
E4	INTC Priority Select Register (INTC_PSR66)	16	R/W	8000h	28.5.12/ 1055
E6	INTC Priority Select Register (INTC_PSR67)	16	R/W	8000h	28.5.12/ 1055
E8	INTC Priority Select Register (INTC_PSR68)	16	R/W	8000h	28.5.12/ 1055
EA	INTC Priority Select Register (INTC_PSR69)	16	R/W	8000h	28.5.12/ 1055
EC	INTC Priority Select Register (INTC_PSR70)	16	R/W	8000h	28.5.12/ 1055
EE	INTC Priority Select Register (INTC_PSR71)	16	R/W	8000h	28.5.12/ 1055
F0	INTC Priority Select Register (INTC_PSR72)	16	R/W	8000h	28.5.12/ 1055
F2	INTC Priority Select Register (INTC_PSR73)	16	R/W	8000h	28.5.12/ 1055
F4	INTC Priority Select Register (INTC_PSR74)	16	R/W	8000h	28.5.12/ 1055
F6	INTC Priority Select Register (INTC_PSR75)	16	R/W	8000h	28.5.12/ 1055
F8	INTC Priority Select Register (INTC_PSR76)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FA	INTC Priority Select Register (INTC_PSR77)	16	R/W	8000h	28.5.12/ 1055
FC	INTC Priority Select Register (INTC_PSR78)	16	R/W	8000h	28.5.12/ 1055
FE	INTC Priority Select Register (INTC_PSR79)	16	R/W	8000h	28.5.12/ 1055
100	INTC Priority Select Register (INTC_PSR80)	16	R/W	8000h	28.5.12/ 1055
102	INTC Priority Select Register (INTC_PSR81)	16	R/W	8000h	28.5.12/ 1055
104	INTC Priority Select Register (INTC_PSR82)	16	R/W	8000h	28.5.12/ 1055
106	INTC Priority Select Register (INTC_PSR83)	16	R/W	8000h	28.5.12/ 1055
108	INTC Priority Select Register (INTC_PSR84)	16	R/W	8000h	28.5.12/ 1055
10A	INTC Priority Select Register (INTC_PSR85)	16	R/W	8000h	28.5.12/ 1055
10C	INTC Priority Select Register (INTC_PSR86)	16	R/W	8000h	28.5.12/ 1055
10E	INTC Priority Select Register (INTC_PSR87)	16	R/W	8000h	28.5.12/ 1055
110	INTC Priority Select Register (INTC_PSR88)	16	R/W	8000h	28.5.12/ 1055
112	INTC Priority Select Register (INTC_PSR89)	16	R/W	8000h	28.5.12/ 1055
114	INTC Priority Select Register (INTC_PSR90)	16	R/W	8000h	28.5.12/ 1055
116	INTC Priority Select Register (INTC_PSR91)	16	R/W	8000h	28.5.12/ 1055
118	INTC Priority Select Register (INTC_PSR92)	16	R/W	8000h	28.5.12/ 1055
11A	INTC Priority Select Register (INTC_PSR93)	16	R/W	8000h	28.5.12/ 1055
11C	INTC Priority Select Register (INTC_PSR94)	16	R/W	8000h	28.5.12/ 1055
11E	INTC Priority Select Register (INTC_PSR95)	16	R/W	8000h	28.5.12/ 1055
120	INTC Priority Select Register (INTC_PSR96)	16	R/W	8000h	28.5.12/ 1055
122	INTC Priority Select Register (INTC_PSR97)	16	R/W	8000h	28.5.12/ 1055
124	INTC Priority Select Register (INTC_PSR98)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
126	INTC Priority Select Register (INTC_PSR99)	16	R/W	8000h	28.5.12/ 1055
128	INTC Priority Select Register (INTC_PSR100)	16	R/W	8000h	28.5.12/ 1055
12A	INTC Priority Select Register (INTC_PSR101)	16	R/W	8000h	28.5.12/ 1055
12C	INTC Priority Select Register (INTC_PSR102)	16	R/W	8000h	28.5.12/ 1055
12E	INTC Priority Select Register (INTC_PSR103)	16	R/W	8000h	28.5.12/ 1055
130	INTC Priority Select Register (INTC_PSR104)	16	R/W	8000h	28.5.12/ 1055
132	INTC Priority Select Register (INTC_PSR105)	16	R/W	8000h	28.5.12/ 1055
134	INTC Priority Select Register (INTC_PSR106)	16	R/W	8000h	28.5.12/ 1055
136	INTC Priority Select Register (INTC_PSR107)	16	R/W	8000h	28.5.12/ 1055
138	INTC Priority Select Register (INTC_PSR108)	16	R/W	8000h	28.5.12/ 1055
13A	INTC Priority Select Register (INTC_PSR109)	16	R/W	8000h	28.5.12/ 1055
13C	INTC Priority Select Register (INTC_PSR110)	16	R/W	8000h	28.5.12/ 1055
13E	INTC Priority Select Register (INTC_PSR111)	16	R/W	8000h	28.5.12/ 1055
140	INTC Priority Select Register (INTC_PSR112)	16	R/W	8000h	28.5.12/ 1055
142	INTC Priority Select Register (INTC_PSR113)	16	R/W	8000h	28.5.12/ 1055
144	INTC Priority Select Register (INTC_PSR114)	16	R/W	8000h	28.5.12/ 1055
146	INTC Priority Select Register (INTC_PSR115)	16	R/W	8000h	28.5.12/ 1055
148	INTC Priority Select Register (INTC_PSR116)	16	R/W	8000h	28.5.12/ 1055
14A	INTC Priority Select Register (INTC_PSR117)	16	R/W	8000h	28.5.12/ 1055
14C	INTC Priority Select Register (INTC_PSR118)	16	R/W	8000h	28.5.12/ 1055
14E	INTC Priority Select Register (INTC_PSR119)	16	R/W	8000h	28.5.12/ 1055
150	INTC Priority Select Register (INTC_PSR120)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
152	INTC Priority Select Register (INTC_PSR121)	16	R/W	8000h	28.5.12/ 1055
154	INTC Priority Select Register (INTC_PSR122)	16	R/W	8000h	28.5.12/ 1055
156	INTC Priority Select Register (INTC_PSR123)	16	R/W	8000h	28.5.12/ 1055
158	INTC Priority Select Register (INTC_PSR124)	16	R/W	8000h	28.5.12/ 1055
15A	INTC Priority Select Register (INTC_PSR125)	16	R/W	8000h	28.5.12/ 1055
15C	INTC Priority Select Register (INTC_PSR126)	16	R/W	8000h	28.5.12/ 1055
15E	INTC Priority Select Register (INTC_PSR127)	16	R/W	8000h	28.5.12/ 1055
160	INTC Priority Select Register (INTC_PSR128)	16	R/W	8000h	28.5.12/ 1055
162	INTC Priority Select Register (INTC_PSR129)	16	R/W	8000h	28.5.12/ 1055
164	INTC Priority Select Register (INTC_PSR130)	16	R/W	8000h	28.5.12/ 1055
166	INTC Priority Select Register (INTC_PSR131)	16	R/W	8000h	28.5.12/ 1055
168	INTC Priority Select Register (INTC_PSR132)	16	R/W	8000h	28.5.12/ 1055
16A	INTC Priority Select Register (INTC_PSR133)	16	R/W	8000h	28.5.12/ 1055
16C	INTC Priority Select Register (INTC_PSR134)	16	R/W	8000h	28.5.12/ 1055
16E	INTC Priority Select Register (INTC_PSR135)	16	R/W	8000h	28.5.12/ 1055
170	INTC Priority Select Register (INTC_PSR136)	16	R/W	8000h	28.5.12/ 1055
172	INTC Priority Select Register (INTC_PSR137)	16	R/W	8000h	28.5.12/ 1055
174	INTC Priority Select Register (INTC_PSR138)	16	R/W	8000h	28.5.12/ 1055
176	INTC Priority Select Register (INTC_PSR139)	16	R/W	8000h	28.5.12/ 1055
178	INTC Priority Select Register (INTC_PSR140)	16	R/W	8000h	28.5.12/ 1055
17A	INTC Priority Select Register (INTC_PSR141)	16	R/W	8000h	28.5.12/ 1055
17C	INTC Priority Select Register (INTC_PSR142)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17E	INTC Priority Select Register (INTC_PSR143)	16	R/W	8000h	28.5.12/ 1055
180	INTC Priority Select Register (INTC_PSR144)	16	R/W	8000h	28.5.12/ 1055
182	INTC Priority Select Register (INTC_PSR145)	16	R/W	8000h	28.5.12/ 1055
184	INTC Priority Select Register (INTC_PSR146)	16	R/W	8000h	28.5.12/ 1055
186	INTC Priority Select Register (INTC_PSR147)	16	R/W	8000h	28.5.12/ 1055
188	INTC Priority Select Register (INTC_PSR148)	16	R/W	8000h	28.5.12/ 1055
18A	INTC Priority Select Register (INTC_PSR149)	16	R/W	8000h	28.5.12/ 1055
18C	INTC Priority Select Register (INTC_PSR150)	16	R/W	8000h	28.5.12/ 1055
18E	INTC Priority Select Register (INTC_PSR151)	16	R/W	8000h	28.5.12/ 1055
190	INTC Priority Select Register (INTC_PSR152)	16	R/W	8000h	28.5.12/ 1055
192	INTC Priority Select Register (INTC_PSR153)	16	R/W	8000h	28.5.12/ 1055
194	INTC Priority Select Register (INTC_PSR154)	16	R/W	8000h	28.5.12/ 1055
196	INTC Priority Select Register (INTC_PSR155)	16	R/W	8000h	28.5.12/ 1055
198	INTC Priority Select Register (INTC_PSR156)	16	R/W	8000h	28.5.12/ 1055
19A	INTC Priority Select Register (INTC_PSR157)	16	R/W	8000h	28.5.12/ 1055
19C	INTC Priority Select Register (INTC_PSR158)	16	R/W	8000h	28.5.12/ 1055
19E	INTC Priority Select Register (INTC_PSR159)	16	R/W	8000h	28.5.12/ 1055
1A0	INTC Priority Select Register (INTC_PSR160)	16	R/W	8000h	28.5.12/ 1055
1A2	INTC Priority Select Register (INTC_PSR161)	16	R/W	8000h	28.5.12/ 1055
1A4	INTC Priority Select Register (INTC_PSR162)	16	R/W	8000h	28.5.12/ 1055
1A6	INTC Priority Select Register (INTC_PSR163)	16	R/W	8000h	28.5.12/ 1055
1A8	INTC Priority Select Register (INTC_PSR164)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1AA	INTC Priority Select Register (INTC_PSR165)	16	R/W	8000h	28.5.12/ 1055
1AC	INTC Priority Select Register (INTC_PSR166)	16	R/W	8000h	28.5.12/ 1055
1AE	INTC Priority Select Register (INTC_PSR167)	16	R/W	8000h	28.5.12/ 1055
1B0	INTC Priority Select Register (INTC_PSR168)	16	R/W	8000h	28.5.12/ 1055
1B2	INTC Priority Select Register (INTC_PSR169)	16	R/W	8000h	28.5.12/ 1055
1B4	INTC Priority Select Register (INTC_PSR170)	16	R/W	8000h	28.5.12/ 1055
1B6	INTC Priority Select Register (INTC_PSR171)	16	R/W	8000h	28.5.12/ 1055
1B8	INTC Priority Select Register (INTC_PSR172)	16	R/W	8000h	28.5.12/ 1055
1BA	INTC Priority Select Register (INTC_PSR173)	16	R/W	8000h	28.5.12/ 1055
1BC	INTC Priority Select Register (INTC_PSR174)	16	R/W	8000h	28.5.12/ 1055
1BE	INTC Priority Select Register (INTC_PSR175)	16	R/W	8000h	28.5.12/ 1055
1C0	INTC Priority Select Register (INTC_PSR176)	16	R/W	8000h	28.5.12/ 1055
1C2	INTC Priority Select Register (INTC_PSR177)	16	R/W	8000h	28.5.12/ 1055
1C4	INTC Priority Select Register (INTC_PSR178)	16	R/W	8000h	28.5.12/ 1055
1C6	INTC Priority Select Register (INTC_PSR179)	16	R/W	8000h	28.5.12/ 1055
1C8	INTC Priority Select Register (INTC_PSR180)	16	R/W	8000h	28.5.12/ 1055
1CA	INTC Priority Select Register (INTC_PSR181)	16	R/W	8000h	28.5.12/ 1055
1CC	INTC Priority Select Register (INTC_PSR182)	16	R/W	8000h	28.5.12/ 1055
1CE	INTC Priority Select Register (INTC_PSR183)	16	R/W	8000h	28.5.12/ 1055
1D0	INTC Priority Select Register (INTC_PSR184)	16	R/W	8000h	28.5.12/ 1055
1D2	INTC Priority Select Register (INTC_PSR185)	16	R/W	8000h	28.5.12/ 1055
1D4	INTC Priority Select Register (INTC_PSR186)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1D6	INTC Priority Select Register (INTC_PSR187)	16	R/W	8000h	28.5.12/ 1055
1D8	INTC Priority Select Register (INTC_PSR188)	16	R/W	8000h	28.5.12/ 1055
1DA	INTC Priority Select Register (INTC_PSR189)	16	R/W	8000h	28.5.12/ 1055
1DC	INTC Priority Select Register (INTC_PSR190)	16	R/W	8000h	28.5.12/ 1055
1DE	INTC Priority Select Register (INTC_PSR191)	16	R/W	8000h	28.5.12/ 1055
1E0	INTC Priority Select Register (INTC_PSR192)	16	R/W	8000h	28.5.12/ 1055
1E2	INTC Priority Select Register (INTC_PSR193)	16	R/W	8000h	28.5.12/ 1055
1E4	INTC Priority Select Register (INTC_PSR194)	16	R/W	8000h	28.5.12/ 1055
1E6	INTC Priority Select Register (INTC_PSR195)	16	R/W	8000h	28.5.12/ 1055
1E8	INTC Priority Select Register (INTC_PSR196)	16	R/W	8000h	28.5.12/ 1055
1EA	INTC Priority Select Register (INTC_PSR197)	16	R/W	8000h	28.5.12/ 1055
1EC	INTC Priority Select Register (INTC_PSR198)	16	R/W	8000h	28.5.12/ 1055
1EE	INTC Priority Select Register (INTC_PSR199)	16	R/W	8000h	28.5.12/ 1055
1F0	INTC Priority Select Register (INTC_PSR200)	16	R/W	8000h	28.5.12/ 1055
1F2	INTC Priority Select Register (INTC_PSR201)	16	R/W	8000h	28.5.12/ 1055
1F4	INTC Priority Select Register (INTC_PSR202)	16	R/W	8000h	28.5.12/ 1055
1F6	INTC Priority Select Register (INTC_PSR203)	16	R/W	8000h	28.5.12/ 1055
1F8	INTC Priority Select Register (INTC_PSR204)	16	R/W	8000h	28.5.12/ 1055
1FA	INTC Priority Select Register (INTC_PSR205)	16	R/W	8000h	28.5.12/ 1055
1FC	INTC Priority Select Register (INTC_PSR206)	16	R/W	8000h	28.5.12/ 1055
1FE	INTC Priority Select Register (INTC_PSR207)	16	R/W	8000h	28.5.12/ 1055
200	INTC Priority Select Register (INTC_PSR208)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202	INTC Priority Select Register (INTC_PSR209)	16	R/W	8000h	28.5.12/ 1055
204	INTC Priority Select Register (INTC_PSR210)	16	R/W	8000h	28.5.12/ 1055
206	INTC Priority Select Register (INTC_PSR211)	16	R/W	8000h	28.5.12/ 1055
208	INTC Priority Select Register (INTC_PSR212)	16	R/W	8000h	28.5.12/ 1055
20A	INTC Priority Select Register (INTC_PSR213)	16	R/W	8000h	28.5.12/ 1055
20C	INTC Priority Select Register (INTC_PSR214)	16	R/W	8000h	28.5.12/ 1055
20E	INTC Priority Select Register (INTC_PSR215)	16	R/W	8000h	28.5.12/ 1055
210	INTC Priority Select Register (INTC_PSR216)	16	R/W	8000h	28.5.12/ 1055
212	INTC Priority Select Register (INTC_PSR217)	16	R/W	8000h	28.5.12/ 1055
214	INTC Priority Select Register (INTC_PSR218)	16	R/W	8000h	28.5.12/ 1055
216	INTC Priority Select Register (INTC_PSR219)	16	R/W	8000h	28.5.12/ 1055
218	INTC Priority Select Register (INTC_PSR220)	16	R/W	8000h	28.5.12/ 1055
21A	INTC Priority Select Register (INTC_PSR221)	16	R/W	8000h	28.5.12/ 1055
21C	INTC Priority Select Register (INTC_PSR222)	16	R/W	8000h	28.5.12/ 1055
21E	INTC Priority Select Register (INTC_PSR223)	16	R/W	8000h	28.5.12/ 1055
220	INTC Priority Select Register (INTC_PSR224)	16	R/W	8000h	28.5.12/ 1055
222	INTC Priority Select Register (INTC_PSR225)	16	R/W	8000h	28.5.12/ 1055
224	INTC Priority Select Register (INTC_PSR226)	16	R/W	8000h	28.5.12/ 1055
226	INTC Priority Select Register (INTC_PSR227)	16	R/W	8000h	28.5.12/ 1055
228	INTC Priority Select Register (INTC_PSR228)	16	R/W	8000h	28.5.12/ 1055
22A	INTC Priority Select Register (INTC_PSR229)	16	R/W	8000h	28.5.12/ 1055
22C	INTC Priority Select Register (INTC_PSR230)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
22E	INTC Priority Select Register (INTC_PSR231)	16	R/W	8000h	28.5.12/ 1055
230	INTC Priority Select Register (INTC_PSR232)	16	R/W	8000h	28.5.12/ 1055
232	INTC Priority Select Register (INTC_PSR233)	16	R/W	8000h	28.5.12/ 1055
234	INTC Priority Select Register (INTC_PSR234)	16	R/W	8000h	28.5.12/ 1055
236	INTC Priority Select Register (INTC_PSR235)	16	R/W	8000h	28.5.12/ 1055
238	INTC Priority Select Register (INTC_PSR236)	16	R/W	8000h	28.5.12/ 1055
23A	INTC Priority Select Register (INTC_PSR237)	16	R/W	8000h	28.5.12/ 1055
23C	INTC Priority Select Register (INTC_PSR238)	16	R/W	8000h	28.5.12/ 1055
23E	INTC Priority Select Register (INTC_PSR239)	16	R/W	8000h	28.5.12/ 1055
240	INTC Priority Select Register (INTC_PSR240)	16	R/W	8000h	28.5.12/ 1055
242	INTC Priority Select Register (INTC_PSR241)	16	R/W	8000h	28.5.12/ 1055
244	INTC Priority Select Register (INTC_PSR242)	16	R/W	8000h	28.5.12/ 1055
246	INTC Priority Select Register (INTC_PSR243)	16	R/W	8000h	28.5.12/ 1055
248	INTC Priority Select Register (INTC_PSR244)	16	R/W	8000h	28.5.12/ 1055
24A	INTC Priority Select Register (INTC_PSR245)	16	R/W	8000h	28.5.12/ 1055
24C	INTC Priority Select Register (INTC_PSR246)	16	R/W	8000h	28.5.12/ 1055
24E	INTC Priority Select Register (INTC_PSR247)	16	R/W	8000h	28.5.12/ 1055
250	INTC Priority Select Register (INTC_PSR248)	16	R/W	8000h	28.5.12/ 1055
252	INTC Priority Select Register (INTC_PSR249)	16	R/W	8000h	28.5.12/ 1055
254	INTC Priority Select Register (INTC_PSR250)	16	R/W	8000h	28.5.12/ 1055
256	INTC Priority Select Register (INTC_PSR251)	16	R/W	8000h	28.5.12/ 1055
258	INTC Priority Select Register (INTC_PSR252)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
25A	INTC Priority Select Register (INTC_PSR253)	16	R/W	8000h	28.5.12/ 1055
25C	INTC Priority Select Register (INTC_PSR254)	16	R/W	8000h	28.5.12/ 1055
25E	INTC Priority Select Register (INTC_PSR255)	16	R/W	8000h	28.5.12/ 1055
260	INTC Priority Select Register (INTC_PSR256)	16	R/W	8000h	28.5.12/ 1055
262	INTC Priority Select Register (INTC_PSR257)	16	R/W	8000h	28.5.12/ 1055
264	INTC Priority Select Register (INTC_PSR258)	16	R/W	8000h	28.5.12/ 1055
266	INTC Priority Select Register (INTC_PSR259)	16	R/W	8000h	28.5.12/ 1055
268	INTC Priority Select Register (INTC_PSR260)	16	R/W	8000h	28.5.12/ 1055
26A	INTC Priority Select Register (INTC_PSR261)	16	R/W	8000h	28.5.12/ 1055
26C	INTC Priority Select Register (INTC_PSR262)	16	R/W	8000h	28.5.12/ 1055
26E	INTC Priority Select Register (INTC_PSR263)	16	R/W	8000h	28.5.12/ 1055
270	INTC Priority Select Register (INTC_PSR264)	16	R/W	8000h	28.5.12/ 1055
272	INTC Priority Select Register (INTC_PSR265)	16	R/W	8000h	28.5.12/ 1055
274	INTC Priority Select Register (INTC_PSR266)	16	R/W	8000h	28.5.12/ 1055
276	INTC Priority Select Register (INTC_PSR267)	16	R/W	8000h	28.5.12/ 1055
278	INTC Priority Select Register (INTC_PSR268)	16	R/W	8000h	28.5.12/ 1055
27A	INTC Priority Select Register (INTC_PSR269)	16	R/W	8000h	28.5.12/ 1055
27C	INTC Priority Select Register (INTC_PSR270)	16	R/W	8000h	28.5.12/ 1055
27E	INTC Priority Select Register (INTC_PSR271)	16	R/W	8000h	28.5.12/ 1055
280	INTC Priority Select Register (INTC_PSR272)	16	R/W	8000h	28.5.12/ 1055
282	INTC Priority Select Register (INTC_PSR273)	16	R/W	8000h	28.5.12/ 1055
284	INTC Priority Select Register (INTC_PSR274)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
286	INTC Priority Select Register (INTC_PSR275)	16	R/W	8000h	28.5.12/ 1055
288	INTC Priority Select Register (INTC_PSR276)	16	R/W	8000h	28.5.12/ 1055
28A	INTC Priority Select Register (INTC_PSR277)	16	R/W	8000h	28.5.12/ 1055
28C	INTC Priority Select Register (INTC_PSR278)	16	R/W	8000h	28.5.12/ 1055
28E	INTC Priority Select Register (INTC_PSR279)	16	R/W	8000h	28.5.12/ 1055
290	INTC Priority Select Register (INTC_PSR280)	16	R/W	8000h	28.5.12/ 1055
292	INTC Priority Select Register (INTC_PSR281)	16	R/W	8000h	28.5.12/ 1055
294	INTC Priority Select Register (INTC_PSR282)	16	R/W	8000h	28.5.12/ 1055
296	INTC Priority Select Register (INTC_PSR283)	16	R/W	8000h	28.5.12/ 1055
298	INTC Priority Select Register (INTC_PSR284)	16	R/W	8000h	28.5.12/ 1055
29A	INTC Priority Select Register (INTC_PSR285)	16	R/W	8000h	28.5.12/ 1055
29C	INTC Priority Select Register (INTC_PSR286)	16	R/W	8000h	28.5.12/ 1055
29E	INTC Priority Select Register (INTC_PSR287)	16	R/W	8000h	28.5.12/ 1055
2A0	INTC Priority Select Register (INTC_PSR288)	16	R/W	8000h	28.5.12/ 1055
2A2	INTC Priority Select Register (INTC_PSR289)	16	R/W	8000h	28.5.12/ 1055
2A4	INTC Priority Select Register (INTC_PSR290)	16	R/W	8000h	28.5.12/ 1055
2A6	INTC Priority Select Register (INTC_PSR291)	16	R/W	8000h	28.5.12/ 1055
2A8	INTC Priority Select Register (INTC_PSR292)	16	R/W	8000h	28.5.12/ 1055
2AA	INTC Priority Select Register (INTC_PSR293)	16	R/W	8000h	28.5.12/ 1055
2AC	INTC Priority Select Register (INTC_PSR294)	16	R/W	8000h	28.5.12/ 1055
2AE	INTC Priority Select Register (INTC_PSR295)	16	R/W	8000h	28.5.12/ 1055
2B0	INTC Priority Select Register (INTC_PSR296)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B2	INTC Priority Select Register (INTC_PSR297)	16	R/W	8000h	28.5.12/ 1055
2B4	INTC Priority Select Register (INTC_PSR298)	16	R/W	8000h	28.5.12/ 1055
2B6	INTC Priority Select Register (INTC_PSR299)	16	R/W	8000h	28.5.12/ 1055
2B8	INTC Priority Select Register (INTC_PSR300)	16	R/W	8000h	28.5.12/ 1055
2BA	INTC Priority Select Register (INTC_PSR301)	16	R/W	8000h	28.5.12/ 1055
2BC	INTC Priority Select Register (INTC_PSR302)	16	R/W	8000h	28.5.12/ 1055
2BE	INTC Priority Select Register (INTC_PSR303)	16	R/W	8000h	28.5.12/ 1055
2C0	INTC Priority Select Register (INTC_PSR304)	16	R/W	8000h	28.5.12/ 1055
2C2	INTC Priority Select Register (INTC_PSR305)	16	R/W	8000h	28.5.12/ 1055
2C4	INTC Priority Select Register (INTC_PSR306)	16	R/W	8000h	28.5.12/ 1055
2C6	INTC Priority Select Register (INTC_PSR307)	16	R/W	8000h	28.5.12/ 1055
2C8	INTC Priority Select Register (INTC_PSR308)	16	R/W	8000h	28.5.12/ 1055
2CA	INTC Priority Select Register (INTC_PSR309)	16	R/W	8000h	28.5.12/ 1055
2CC	INTC Priority Select Register (INTC_PSR310)	16	R/W	8000h	28.5.12/ 1055
2CE	INTC Priority Select Register (INTC_PSR311)	16	R/W	8000h	28.5.12/ 1055
2D0	INTC Priority Select Register (INTC_PSR312)	16	R/W	8000h	28.5.12/ 1055
2D2	INTC Priority Select Register (INTC_PSR313)	16	R/W	8000h	28.5.12/ 1055
2D4	INTC Priority Select Register (INTC_PSR314)	16	R/W	8000h	28.5.12/ 1055
2D6	INTC Priority Select Register (INTC_PSR315)	16	R/W	8000h	28.5.12/ 1055
2D8	INTC Priority Select Register (INTC_PSR316)	16	R/W	8000h	28.5.12/ 1055
2DA	INTC Priority Select Register (INTC_PSR317)	16	R/W	8000h	28.5.12/ 1055
2DC	INTC Priority Select Register (INTC_PSR318)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2DE	INTC Priority Select Register (INTC_PSR319)	16	R/W	8000h	28.5.12/ 1055
2E0	INTC Priority Select Register (INTC_PSR320)	16	R/W	8000h	28.5.12/ 1055
2E2	INTC Priority Select Register (INTC_PSR321)	16	R/W	8000h	28.5.12/ 1055
2E4	INTC Priority Select Register (INTC_PSR322)	16	R/W	8000h	28.5.12/ 1055
2E6	INTC Priority Select Register (INTC_PSR323)	16	R/W	8000h	28.5.12/ 1055
2E8	INTC Priority Select Register (INTC_PSR324)	16	R/W	8000h	28.5.12/ 1055
2EA	INTC Priority Select Register (INTC_PSR325)	16	R/W	8000h	28.5.12/ 1055
2EC	INTC Priority Select Register (INTC_PSR326)	16	R/W	8000h	28.5.12/ 1055
2EE	INTC Priority Select Register (INTC_PSR327)	16	R/W	8000h	28.5.12/ 1055
2F0	INTC Priority Select Register (INTC_PSR328)	16	R/W	8000h	28.5.12/ 1055
2F2	INTC Priority Select Register (INTC_PSR329)	16	R/W	8000h	28.5.12/ 1055
2F4	INTC Priority Select Register (INTC_PSR330)	16	R/W	8000h	28.5.12/ 1055
2F6	INTC Priority Select Register (INTC_PSR331)	16	R/W	8000h	28.5.12/ 1055
2F8	INTC Priority Select Register (INTC_PSR332)	16	R/W	8000h	28.5.12/ 1055
2FA	INTC Priority Select Register (INTC_PSR333)	16	R/W	8000h	28.5.12/ 1055
2FC	INTC Priority Select Register (INTC_PSR334)	16	R/W	8000h	28.5.12/ 1055
2FE	INTC Priority Select Register (INTC_PSR335)	16	R/W	8000h	28.5.12/ 1055
300	INTC Priority Select Register (INTC_PSR336)	16	R/W	8000h	28.5.12/ 1055
302	INTC Priority Select Register (INTC_PSR337)	16	R/W	8000h	28.5.12/ 1055
304	INTC Priority Select Register (INTC_PSR338)	16	R/W	8000h	28.5.12/ 1055
306	INTC Priority Select Register (INTC_PSR339)	16	R/W	8000h	28.5.12/ 1055
308	INTC Priority Select Register (INTC_PSR340)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A	INTC Priority Select Register (INTC_PSR341)	16	R/W	8000h	28.5.12/ 1055
30C	INTC Priority Select Register (INTC_PSR342)	16	R/W	8000h	28.5.12/ 1055
30E	INTC Priority Select Register (INTC_PSR343)	16	R/W	8000h	28.5.12/ 1055
310	INTC Priority Select Register (INTC_PSR344)	16	R/W	8000h	28.5.12/ 1055
312	INTC Priority Select Register (INTC_PSR345)	16	R/W	8000h	28.5.12/ 1055
314	INTC Priority Select Register (INTC_PSR346)	16	R/W	8000h	28.5.12/ 1055
316	INTC Priority Select Register (INTC_PSR347)	16	R/W	8000h	28.5.12/ 1055
318	INTC Priority Select Register (INTC_PSR348)	16	R/W	8000h	28.5.12/ 1055
31A	INTC Priority Select Register (INTC_PSR349)	16	R/W	8000h	28.5.12/ 1055
31C	INTC Priority Select Register (INTC_PSR350)	16	R/W	8000h	28.5.12/ 1055
31E	INTC Priority Select Register (INTC_PSR351)	16	R/W	8000h	28.5.12/ 1055
320	INTC Priority Select Register (INTC_PSR352)	16	R/W	8000h	28.5.12/ 1055
322	INTC Priority Select Register (INTC_PSR353)	16	R/W	8000h	28.5.12/ 1055
324	INTC Priority Select Register (INTC_PSR354)	16	R/W	8000h	28.5.12/ 1055
326	INTC Priority Select Register (INTC_PSR355)	16	R/W	8000h	28.5.12/ 1055
328	INTC Priority Select Register (INTC_PSR356)	16	R/W	8000h	28.5.12/ 1055
32A	INTC Priority Select Register (INTC_PSR357)	16	R/W	8000h	28.5.12/ 1055
32C	INTC Priority Select Register (INTC_PSR358)	16	R/W	8000h	28.5.12/ 1055
32E	INTC Priority Select Register (INTC_PSR359)	16	R/W	8000h	28.5.12/ 1055
330	INTC Priority Select Register (INTC_PSR360)	16	R/W	8000h	28.5.12/ 1055
332	INTC Priority Select Register (INTC_PSR361)	16	R/W	8000h	28.5.12/ 1055
334	INTC Priority Select Register (INTC_PSR362)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
336	INTC Priority Select Register (INTC_PSR363)	16	R/W	8000h	28.5.12/ 1055
338	INTC Priority Select Register (INTC_PSR364)	16	R/W	8000h	28.5.12/ 1055
33A	INTC Priority Select Register (INTC_PSR365)	16	R/W	8000h	28.5.12/ 1055
33C	INTC Priority Select Register (INTC_PSR366)	16	R/W	8000h	28.5.12/ 1055
33E	INTC Priority Select Register (INTC_PSR367)	16	R/W	8000h	28.5.12/ 1055
340	INTC Priority Select Register (INTC_PSR368)	16	R/W	8000h	28.5.12/ 1055
342	INTC Priority Select Register (INTC_PSR369)	16	R/W	8000h	28.5.12/ 1055
344	INTC Priority Select Register (INTC_PSR370)	16	R/W	8000h	28.5.12/ 1055
346	INTC Priority Select Register (INTC_PSR371)	16	R/W	8000h	28.5.12/ 1055
348	INTC Priority Select Register (INTC_PSR372)	16	R/W	8000h	28.5.12/ 1055
34A	INTC Priority Select Register (INTC_PSR373)	16	R/W	8000h	28.5.12/ 1055
34C	INTC Priority Select Register (INTC_PSR374)	16	R/W	8000h	28.5.12/ 1055
34E	INTC Priority Select Register (INTC_PSR375)	16	R/W	8000h	28.5.12/ 1055
350	INTC Priority Select Register (INTC_PSR376)	16	R/W	8000h	28.5.12/ 1055
352	INTC Priority Select Register (INTC_PSR377)	16	R/W	8000h	28.5.12/ 1055
354	INTC Priority Select Register (INTC_PSR378)	16	R/W	8000h	28.5.12/ 1055
356	INTC Priority Select Register (INTC_PSR379)	16	R/W	8000h	28.5.12/ 1055
358	INTC Priority Select Register (INTC_PSR380)	16	R/W	8000h	28.5.12/ 1055
35A	INTC Priority Select Register (INTC_PSR381)	16	R/W	8000h	28.5.12/ 1055
35C	INTC Priority Select Register (INTC_PSR382)	16	R/W	8000h	28.5.12/ 1055
35E	INTC Priority Select Register (INTC_PSR383)	16	R/W	8000h	28.5.12/ 1055
360	INTC Priority Select Register (INTC_PSR384)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
362	INTC Priority Select Register (INTC_PSR385)	16	R/W	8000h	28.5.12/ 1055
364	INTC Priority Select Register (INTC_PSR386)	16	R/W	8000h	28.5.12/ 1055
366	INTC Priority Select Register (INTC_PSR387)	16	R/W	8000h	28.5.12/ 1055
368	INTC Priority Select Register (INTC_PSR388)	16	R/W	8000h	28.5.12/ 1055
36A	INTC Priority Select Register (INTC_PSR389)	16	R/W	8000h	28.5.12/ 1055
36C	INTC Priority Select Register (INTC_PSR390)	16	R/W	8000h	28.5.12/ 1055
36E	INTC Priority Select Register (INTC_PSR391)	16	R/W	8000h	28.5.12/ 1055
370	INTC Priority Select Register (INTC_PSR392)	16	R/W	8000h	28.5.12/ 1055
372	INTC Priority Select Register (INTC_PSR393)	16	R/W	8000h	28.5.12/ 1055
374	INTC Priority Select Register (INTC_PSR394)	16	R/W	8000h	28.5.12/ 1055
376	INTC Priority Select Register (INTC_PSR395)	16	R/W	8000h	28.5.12/ 1055
378	INTC Priority Select Register (INTC_PSR396)	16	R/W	8000h	28.5.12/ 1055
37A	INTC Priority Select Register (INTC_PSR397)	16	R/W	8000h	28.5.12/ 1055
37C	INTC Priority Select Register (INTC_PSR398)	16	R/W	8000h	28.5.12/ 1055
37E	INTC Priority Select Register (INTC_PSR399)	16	R/W	8000h	28.5.12/ 1055
380	INTC Priority Select Register (INTC_PSR400)	16	R/W	8000h	28.5.12/ 1055
382	INTC Priority Select Register (INTC_PSR401)	16	R/W	8000h	28.5.12/ 1055
384	INTC Priority Select Register (INTC_PSR402)	16	R/W	8000h	28.5.12/ 1055
386	INTC Priority Select Register (INTC_PSR403)	16	R/W	8000h	28.5.12/ 1055
388	INTC Priority Select Register (INTC_PSR404)	16	R/W	8000h	28.5.12/ 1055
38A	INTC Priority Select Register (INTC_PSR405)	16	R/W	8000h	28.5.12/ 1055
38C	INTC Priority Select Register (INTC_PSR406)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38E	INTC Priority Select Register (INTC_PSR407)	16	R/W	8000h	28.5.12/ 1055
390	INTC Priority Select Register (INTC_PSR408)	16	R/W	8000h	28.5.12/ 1055
392	INTC Priority Select Register (INTC_PSR409)	16	R/W	8000h	28.5.12/ 1055
394	INTC Priority Select Register (INTC_PSR410)	16	R/W	8000h	28.5.12/ 1055
396	INTC Priority Select Register (INTC_PSR411)	16	R/W	8000h	28.5.12/ 1055
398	INTC Priority Select Register (INTC_PSR412)	16	R/W	8000h	28.5.12/ 1055
39A	INTC Priority Select Register (INTC_PSR413)	16	R/W	8000h	28.5.12/ 1055
39C	INTC Priority Select Register (INTC_PSR414)	16	R/W	8000h	28.5.12/ 1055
39E	INTC Priority Select Register (INTC_PSR415)	16	R/W	8000h	28.5.12/ 1055
3A0	INTC Priority Select Register (INTC_PSR416)	16	R/W	8000h	28.5.12/ 1055
3A2	INTC Priority Select Register (INTC_PSR417)	16	R/W	8000h	28.5.12/ 1055
3A4	INTC Priority Select Register (INTC_PSR418)	16	R/W	8000h	28.5.12/ 1055
3A6	INTC Priority Select Register (INTC_PSR419)	16	R/W	8000h	28.5.12/ 1055
3A8	INTC Priority Select Register (INTC_PSR420)	16	R/W	8000h	28.5.12/ 1055
3AA	INTC Priority Select Register (INTC_PSR421)	16	R/W	8000h	28.5.12/ 1055
3AC	INTC Priority Select Register (INTC_PSR422)	16	R/W	8000h	28.5.12/ 1055
3AE	INTC Priority Select Register (INTC_PSR423)	16	R/W	8000h	28.5.12/ 1055
3B0	INTC Priority Select Register (INTC_PSR424)	16	R/W	8000h	28.5.12/ 1055
3B2	INTC Priority Select Register (INTC_PSR425)	16	R/W	8000h	28.5.12/ 1055
3B4	INTC Priority Select Register (INTC_PSR426)	16	R/W	8000h	28.5.12/ 1055
3B6	INTC Priority Select Register (INTC_PSR427)	16	R/W	8000h	28.5.12/ 1055
3B8	INTC Priority Select Register (INTC_PSR428)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3BA	INTC Priority Select Register (INTC_PSR429)	16	R/W	8000h	28.5.12/ 1055
3BC	INTC Priority Select Register (INTC_PSR430)	16	R/W	8000h	28.5.12/ 1055
3BE	INTC Priority Select Register (INTC_PSR431)	16	R/W	8000h	28.5.12/ 1055
3C0	INTC Priority Select Register (INTC_PSR432)	16	R/W	8000h	28.5.12/ 1055
3C2	INTC Priority Select Register (INTC_PSR433)	16	R/W	8000h	28.5.12/ 1055
3C4	INTC Priority Select Register (INTC_PSR434)	16	R/W	8000h	28.5.12/ 1055
3C6	INTC Priority Select Register (INTC_PSR435)	16	R/W	8000h	28.5.12/ 1055
3C8	INTC Priority Select Register (INTC_PSR436)	16	R/W	8000h	28.5.12/ 1055
3CA	INTC Priority Select Register (INTC_PSR437)	16	R/W	8000h	28.5.12/ 1055
3CC	INTC Priority Select Register (INTC_PSR438)	16	R/W	8000h	28.5.12/ 1055
3CE	INTC Priority Select Register (INTC_PSR439)	16	R/W	8000h	28.5.12/ 1055
3D0	INTC Priority Select Register (INTC_PSR440)	16	R/W	8000h	28.5.12/ 1055
3D2	INTC Priority Select Register (INTC_PSR441)	16	R/W	8000h	28.5.12/ 1055
3D4	INTC Priority Select Register (INTC_PSR442)	16	R/W	8000h	28.5.12/ 1055
3D6	INTC Priority Select Register (INTC_PSR443)	16	R/W	8000h	28.5.12/ 1055
3D8	INTC Priority Select Register (INTC_PSR444)	16	R/W	8000h	28.5.12/ 1055
3DA	INTC Priority Select Register (INTC_PSR445)	16	R/W	8000h	28.5.12/ 1055
3DC	INTC Priority Select Register (INTC_PSR446)	16	R/W	8000h	28.5.12/ 1055
3DE	INTC Priority Select Register (INTC_PSR447)	16	R/W	8000h	28.5.12/ 1055
3E0	INTC Priority Select Register (INTC_PSR448)	16	R/W	8000h	28.5.12/ 1055
3E2	INTC Priority Select Register (INTC_PSR449)	16	R/W	8000h	28.5.12/ 1055
3E4	INTC Priority Select Register (INTC_PSR450)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3E6	INTC Priority Select Register (INTC_PSR451)	16	R/W	8000h	28.5.12/ 1055
3E8	INTC Priority Select Register (INTC_PSR452)	16	R/W	8000h	28.5.12/ 1055
3EA	INTC Priority Select Register (INTC_PSR453)	16	R/W	8000h	28.5.12/ 1055
3EC	INTC Priority Select Register (INTC_PSR454)	16	R/W	8000h	28.5.12/ 1055
3EE	INTC Priority Select Register (INTC_PSR455)	16	R/W	8000h	28.5.12/ 1055
3F0	INTC Priority Select Register (INTC_PSR456)	16	R/W	8000h	28.5.12/ 1055
3F2	INTC Priority Select Register (INTC_PSR457)	16	R/W	8000h	28.5.12/ 1055
3F4	INTC Priority Select Register (INTC_PSR458)	16	R/W	8000h	28.5.12/ 1055
3F6	INTC Priority Select Register (INTC_PSR459)	16	R/W	8000h	28.5.12/ 1055
3F8	INTC Priority Select Register (INTC_PSR460)	16	R/W	8000h	28.5.12/ 1055
3FA	INTC Priority Select Register (INTC_PSR461)	16	R/W	8000h	28.5.12/ 1055
3FC	INTC Priority Select Register (INTC_PSR462)	16	R/W	8000h	28.5.12/ 1055
3FE	INTC Priority Select Register (INTC_PSR463)	16	R/W	8000h	28.5.12/ 1055
400	INTC Priority Select Register (INTC_PSR464)	16	R/W	8000h	28.5.12/ 1055
402	INTC Priority Select Register (INTC_PSR465)	16	R/W	8000h	28.5.12/ 1055
404	INTC Priority Select Register (INTC_PSR466)	16	R/W	8000h	28.5.12/ 1055
406	INTC Priority Select Register (INTC_PSR467)	16	R/W	8000h	28.5.12/ 1055
408	INTC Priority Select Register (INTC_PSR468)	16	R/W	8000h	28.5.12/ 1055
40A	INTC Priority Select Register (INTC_PSR469)	16	R/W	8000h	28.5.12/ 1055
40C	INTC Priority Select Register (INTC_PSR470)	16	R/W	8000h	28.5.12/ 1055
40E	INTC Priority Select Register (INTC_PSR471)	16	R/W	8000h	28.5.12/ 1055
410	INTC Priority Select Register (INTC_PSR472)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
412	INTC Priority Select Register (INTC_PSR473)	16	R/W	8000h	28.5.12/ 1055
414	INTC Priority Select Register (INTC_PSR474)	16	R/W	8000h	28.5.12/ 1055
416	INTC Priority Select Register (INTC_PSR475)	16	R/W	8000h	28.5.12/ 1055
418	INTC Priority Select Register (INTC_PSR476)	16	R/W	8000h	28.5.12/ 1055
41A	INTC Priority Select Register (INTC_PSR477)	16	R/W	8000h	28.5.12/ 1055
41C	INTC Priority Select Register (INTC_PSR478)	16	R/W	8000h	28.5.12/ 1055
41E	INTC Priority Select Register (INTC_PSR479)	16	R/W	8000h	28.5.12/ 1055
420	INTC Priority Select Register (INTC_PSR480)	16	R/W	8000h	28.5.12/ 1055
422	INTC Priority Select Register (INTC_PSR481)	16	R/W	8000h	28.5.12/ 1055
424	INTC Priority Select Register (INTC_PSR482)	16	R/W	8000h	28.5.12/ 1055
426	INTC Priority Select Register (INTC_PSR483)	16	R/W	8000h	28.5.12/ 1055
428	INTC Priority Select Register (INTC_PSR484)	16	R/W	8000h	28.5.12/ 1055
42A	INTC Priority Select Register (INTC_PSR485)	16	R/W	8000h	28.5.12/ 1055
42C	INTC Priority Select Register (INTC_PSR486)	16	R/W	8000h	28.5.12/ 1055
42E	INTC Priority Select Register (INTC_PSR487)	16	R/W	8000h	28.5.12/ 1055
430	INTC Priority Select Register (INTC_PSR488)	16	R/W	8000h	28.5.12/ 1055
432	INTC Priority Select Register (INTC_PSR489)	16	R/W	8000h	28.5.12/ 1055
434	INTC Priority Select Register (INTC_PSR490)	16	R/W	8000h	28.5.12/ 1055
436	INTC Priority Select Register (INTC_PSR491)	16	R/W	8000h	28.5.12/ 1055
438	INTC Priority Select Register (INTC_PSR492)	16	R/W	8000h	28.5.12/ 1055
43A	INTC Priority Select Register (INTC_PSR493)	16	R/W	8000h	28.5.12/ 1055
43C	INTC Priority Select Register (INTC_PSR494)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
43E	INTC Priority Select Register (INTC_PSR495)	16	R/W	8000h	28.5.12/ 1055
440	INTC Priority Select Register (INTC_PSR496)	16	R/W	8000h	28.5.12/ 1055
442	INTC Priority Select Register (INTC_PSR497)	16	R/W	8000h	28.5.12/ 1055
444	INTC Priority Select Register (INTC_PSR498)	16	R/W	8000h	28.5.12/ 1055
446	INTC Priority Select Register (INTC_PSR499)	16	R/W	8000h	28.5.12/ 1055
448	INTC Priority Select Register (INTC_PSR500)	16	R/W	8000h	28.5.12/ 1055
44A	INTC Priority Select Register (INTC_PSR501)	16	R/W	8000h	28.5.12/ 1055
44C	INTC Priority Select Register (INTC_PSR502)	16	R/W	8000h	28.5.12/ 1055
44E	INTC Priority Select Register (INTC_PSR503)	16	R/W	8000h	28.5.12/ 1055
450	INTC Priority Select Register (INTC_PSR504)	16	R/W	8000h	28.5.12/ 1055
452	INTC Priority Select Register (INTC_PSR505)	16	R/W	8000h	28.5.12/ 1055
454	INTC Priority Select Register (INTC_PSR506)	16	R/W	8000h	28.5.12/ 1055
456	INTC Priority Select Register (INTC_PSR507)	16	R/W	8000h	28.5.12/ 1055
458	INTC Priority Select Register (INTC_PSR508)	16	R/W	8000h	28.5.12/ 1055
45A	INTC Priority Select Register (INTC_PSR509)	16	R/W	8000h	28.5.12/ 1055
45C	INTC Priority Select Register (INTC_PSR510)	16	R/W	8000h	28.5.12/ 1055
45E	INTC Priority Select Register (INTC_PSR511)	16	R/W	8000h	28.5.12/ 1055
460	INTC Priority Select Register (INTC_PSR512)	16	R/W	8000h	28.5.12/ 1055
462	INTC Priority Select Register (INTC_PSR513)	16	R/W	8000h	28.5.12/ 1055
464	INTC Priority Select Register (INTC_PSR514)	16	R/W	8000h	28.5.12/ 1055
466	INTC Priority Select Register (INTC_PSR515)	16	R/W	8000h	28.5.12/ 1055
468	INTC Priority Select Register (INTC_PSR516)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
46A	INTC Priority Select Register (INTC_PSR517)	16	R/W	8000h	28.5.12/ 1055
46C	INTC Priority Select Register (INTC_PSR518)	16	R/W	8000h	28.5.12/ 1055
46E	INTC Priority Select Register (INTC_PSR519)	16	R/W	8000h	28.5.12/ 1055
470	INTC Priority Select Register (INTC_PSR520)	16	R/W	8000h	28.5.12/ 1055
472	INTC Priority Select Register (INTC_PSR521)	16	R/W	8000h	28.5.12/ 1055
474	INTC Priority Select Register (INTC_PSR522)	16	R/W	8000h	28.5.12/ 1055
476	INTC Priority Select Register (INTC_PSR523)	16	R/W	8000h	28.5.12/ 1055
478	INTC Priority Select Register (INTC_PSR524)	16	R/W	8000h	28.5.12/ 1055
47A	INTC Priority Select Register (INTC_PSR525)	16	R/W	8000h	28.5.12/ 1055
47C	INTC Priority Select Register (INTC_PSR526)	16	R/W	8000h	28.5.12/ 1055
47E	INTC Priority Select Register (INTC_PSR527)	16	R/W	8000h	28.5.12/ 1055
480	INTC Priority Select Register (INTC_PSR528)	16	R/W	8000h	28.5.12/ 1055
482	INTC Priority Select Register (INTC_PSR529)	16	R/W	8000h	28.5.12/ 1055
484	INTC Priority Select Register (INTC_PSR530)	16	R/W	8000h	28.5.12/ 1055
486	INTC Priority Select Register (INTC_PSR531)	16	R/W	8000h	28.5.12/ 1055
488	INTC Priority Select Register (INTC_PSR532)	16	R/W	8000h	28.5.12/ 1055
48A	INTC Priority Select Register (INTC_PSR533)	16	R/W	8000h	28.5.12/ 1055
48C	INTC Priority Select Register (INTC_PSR534)	16	R/W	8000h	28.5.12/ 1055
48E	INTC Priority Select Register (INTC_PSR535)	16	R/W	8000h	28.5.12/ 1055
490	INTC Priority Select Register (INTC_PSR536)	16	R/W	8000h	28.5.12/ 1055
492	INTC Priority Select Register (INTC_PSR537)	16	R/W	8000h	28.5.12/ 1055
494	INTC Priority Select Register (INTC_PSR538)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
496	INTC Priority Select Register (INTC_PSR539)	16	R/W	8000h	28.5.12/ 1055
498	INTC Priority Select Register (INTC_PSR540)	16	R/W	8000h	28.5.12/ 1055
49A	INTC Priority Select Register (INTC_PSR541)	16	R/W	8000h	28.5.12/ 1055
49C	INTC Priority Select Register (INTC_PSR542)	16	R/W	8000h	28.5.12/ 1055
49E	INTC Priority Select Register (INTC_PSR543)	16	R/W	8000h	28.5.12/ 1055
4A0	INTC Priority Select Register (INTC_PSR544)	16	R/W	8000h	28.5.12/ 1055
4A2	INTC Priority Select Register (INTC_PSR545)	16	R/W	8000h	28.5.12/ 1055
4A4	INTC Priority Select Register (INTC_PSR546)	16	R/W	8000h	28.5.12/ 1055
4A6	INTC Priority Select Register (INTC_PSR547)	16	R/W	8000h	28.5.12/ 1055
4A8	INTC Priority Select Register (INTC_PSR548)	16	R/W	8000h	28.5.12/ 1055
4AA	INTC Priority Select Register (INTC_PSR549)	16	R/W	8000h	28.5.12/ 1055
4AC	INTC Priority Select Register (INTC_PSR550)	16	R/W	8000h	28.5.12/ 1055
4AE	INTC Priority Select Register (INTC_PSR551)	16	R/W	8000h	28.5.12/ 1055
4B0	INTC Priority Select Register (INTC_PSR552)	16	R/W	8000h	28.5.12/ 1055
4B2	INTC Priority Select Register (INTC_PSR553)	16	R/W	8000h	28.5.12/ 1055
4B4	INTC Priority Select Register (INTC_PSR554)	16	R/W	8000h	28.5.12/ 1055
4B6	INTC Priority Select Register (INTC_PSR555)	16	R/W	8000h	28.5.12/ 1055
4B8	INTC Priority Select Register (INTC_PSR556)	16	R/W	8000h	28.5.12/ 1055
4BA	INTC Priority Select Register (INTC_PSR557)	16	R/W	8000h	28.5.12/ 1055
4BC	INTC Priority Select Register (INTC_PSR558)	16	R/W	8000h	28.5.12/ 1055
4BE	INTC Priority Select Register (INTC_PSR559)	16	R/W	8000h	28.5.12/ 1055
4C0	INTC Priority Select Register (INTC_PSR560)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4C2	INTC Priority Select Register (INTC_PSR561)	16	R/W	8000h	28.5.12/ 1055
4C4	INTC Priority Select Register (INTC_PSR562)	16	R/W	8000h	28.5.12/ 1055
4C6	INTC Priority Select Register (INTC_PSR563)	16	R/W	8000h	28.5.12/ 1055
4C8	INTC Priority Select Register (INTC_PSR564)	16	R/W	8000h	28.5.12/ 1055
4CA	INTC Priority Select Register (INTC_PSR565)	16	R/W	8000h	28.5.12/ 1055
4CC	INTC Priority Select Register (INTC_PSR566)	16	R/W	8000h	28.5.12/ 1055
4CE	INTC Priority Select Register (INTC_PSR567)	16	R/W	8000h	28.5.12/ 1055
4D0	INTC Priority Select Register (INTC_PSR568)	16	R/W	8000h	28.5.12/ 1055
4D2	INTC Priority Select Register (INTC_PSR569)	16	R/W	8000h	28.5.12/ 1055
4D4	INTC Priority Select Register (INTC_PSR570)	16	R/W	8000h	28.5.12/ 1055
4D6	INTC Priority Select Register (INTC_PSR571)	16	R/W	8000h	28.5.12/ 1055
4D8	INTC Priority Select Register (INTC_PSR572)	16	R/W	8000h	28.5.12/ 1055
4DA	INTC Priority Select Register (INTC_PSR573)	16	R/W	8000h	28.5.12/ 1055
4DC	INTC Priority Select Register (INTC_PSR574)	16	R/W	8000h	28.5.12/ 1055
4DE	INTC Priority Select Register (INTC_PSR575)	16	R/W	8000h	28.5.12/ 1055
4E0	INTC Priority Select Register (INTC_PSR576)	16	R/W	8000h	28.5.12/ 1055
4E2	INTC Priority Select Register (INTC_PSR577)	16	R/W	8000h	28.5.12/ 1055
4E4	INTC Priority Select Register (INTC_PSR578)	16	R/W	8000h	28.5.12/ 1055
4E6	INTC Priority Select Register (INTC_PSR579)	16	R/W	8000h	28.5.12/ 1055
4E8	INTC Priority Select Register (INTC_PSR580)	16	R/W	8000h	28.5.12/ 1055
4EA	INTC Priority Select Register (INTC_PSR581)	16	R/W	8000h	28.5.12/ 1055
4EC	INTC Priority Select Register (INTC_PSR582)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4EE	INTC Priority Select Register (INTC_PSR583)	16	R/W	8000h	28.5.12/ 1055
4F0	INTC Priority Select Register (INTC_PSR584)	16	R/W	8000h	28.5.12/ 1055
4F2	INTC Priority Select Register (INTC_PSR585)	16	R/W	8000h	28.5.12/ 1055
4F4	INTC Priority Select Register (INTC_PSR586)	16	R/W	8000h	28.5.12/ 1055
4F6	INTC Priority Select Register (INTC_PSR587)	16	R/W	8000h	28.5.12/ 1055
4F8	INTC Priority Select Register (INTC_PSR588)	16	R/W	8000h	28.5.12/ 1055
4FA	INTC Priority Select Register (INTC_PSR589)	16	R/W	8000h	28.5.12/ 1055
4FC	INTC Priority Select Register (INTC_PSR590)	16	R/W	8000h	28.5.12/ 1055
4FE	INTC Priority Select Register (INTC_PSR591)	16	R/W	8000h	28.5.12/ 1055
500	INTC Priority Select Register (INTC_PSR592)	16	R/W	8000h	28.5.12/ 1055
502	INTC Priority Select Register (INTC_PSR593)	16	R/W	8000h	28.5.12/ 1055
504	INTC Priority Select Register (INTC_PSR594)	16	R/W	8000h	28.5.12/ 1055
506	INTC Priority Select Register (INTC_PSR595)	16	R/W	8000h	28.5.12/ 1055
508	INTC Priority Select Register (INTC_PSR596)	16	R/W	8000h	28.5.12/ 1055
50A	INTC Priority Select Register (INTC_PSR597)	16	R/W	8000h	28.5.12/ 1055
50C	INTC Priority Select Register (INTC_PSR598)	16	R/W	8000h	28.5.12/ 1055
50E	INTC Priority Select Register (INTC_PSR599)	16	R/W	8000h	28.5.12/ 1055
510	INTC Priority Select Register (INTC_PSR600)	16	R/W	8000h	28.5.12/ 1055
512	INTC Priority Select Register (INTC_PSR601)	16	R/W	8000h	28.5.12/ 1055
514	INTC Priority Select Register (INTC_PSR602)	16	R/W	8000h	28.5.12/ 1055
516	INTC Priority Select Register (INTC_PSR603)	16	R/W	8000h	28.5.12/ 1055
518	INTC Priority Select Register (INTC_PSR604)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
51A	INTC Priority Select Register (INTC_PSR605)	16	R/W	8000h	28.5.12/ 1055
51C	INTC Priority Select Register (INTC_PSR606)	16	R/W	8000h	28.5.12/ 1055
51E	INTC Priority Select Register (INTC_PSR607)	16	R/W	8000h	28.5.12/ 1055
520	INTC Priority Select Register (INTC_PSR608)	16	R/W	8000h	28.5.12/ 1055
522	INTC Priority Select Register (INTC_PSR609)	16	R/W	8000h	28.5.12/ 1055
524	INTC Priority Select Register (INTC_PSR610)	16	R/W	8000h	28.5.12/ 1055
526	INTC Priority Select Register (INTC_PSR611)	16	R/W	8000h	28.5.12/ 1055
528	INTC Priority Select Register (INTC_PSR612)	16	R/W	8000h	28.5.12/ 1055
52A	INTC Priority Select Register (INTC_PSR613)	16	R/W	8000h	28.5.12/ 1055
52C	INTC Priority Select Register (INTC_PSR614)	16	R/W	8000h	28.5.12/ 1055
52E	INTC Priority Select Register (INTC_PSR615)	16	R/W	8000h	28.5.12/ 1055
530	INTC Priority Select Register (INTC_PSR616)	16	R/W	8000h	28.5.12/ 1055
532	INTC Priority Select Register (INTC_PSR617)	16	R/W	8000h	28.5.12/ 1055
534	INTC Priority Select Register (INTC_PSR618)	16	R/W	8000h	28.5.12/ 1055
536	INTC Priority Select Register (INTC_PSR619)	16	R/W	8000h	28.5.12/ 1055
538	INTC Priority Select Register (INTC_PSR620)	16	R/W	8000h	28.5.12/ 1055
53A	INTC Priority Select Register (INTC_PSR621)	16	R/W	8000h	28.5.12/ 1055
53C	INTC Priority Select Register (INTC_PSR622)	16	R/W	8000h	28.5.12/ 1055
53E	INTC Priority Select Register (INTC_PSR623)	16	R/W	8000h	28.5.12/ 1055
540	INTC Priority Select Register (INTC_PSR624)	16	R/W	8000h	28.5.12/ 1055
542	INTC Priority Select Register (INTC_PSR625)	16	R/W	8000h	28.5.12/ 1055
544	INTC Priority Select Register (INTC_PSR626)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
546	INTC Priority Select Register (INTC_PSR627)	16	R/W	8000h	28.5.12/ 1055
548	INTC Priority Select Register (INTC_PSR628)	16	R/W	8000h	28.5.12/ 1055
54A	INTC Priority Select Register (INTC_PSR629)	16	R/W	8000h	28.5.12/ 1055
54C	INTC Priority Select Register (INTC_PSR630)	16	R/W	8000h	28.5.12/ 1055
54E	INTC Priority Select Register (INTC_PSR631)	16	R/W	8000h	28.5.12/ 1055
550	INTC Priority Select Register (INTC_PSR632)	16	R/W	8000h	28.5.12/ 1055
552	INTC Priority Select Register (INTC_PSR633)	16	R/W	8000h	28.5.12/ 1055
554	INTC Priority Select Register (INTC_PSR634)	16	R/W	8000h	28.5.12/ 1055
556	INTC Priority Select Register (INTC_PSR635)	16	R/W	8000h	28.5.12/ 1055
558	INTC Priority Select Register (INTC_PSR636)	16	R/W	8000h	28.5.12/ 1055
55A	INTC Priority Select Register (INTC_PSR637)	16	R/W	8000h	28.5.12/ 1055
55C	INTC Priority Select Register (INTC_PSR638)	16	R/W	8000h	28.5.12/ 1055
55E	INTC Priority Select Register (INTC_PSR639)	16	R/W	8000h	28.5.12/ 1055
560	INTC Priority Select Register (INTC_PSR640)	16	R/W	8000h	28.5.12/ 1055
562	INTC Priority Select Register (INTC_PSR641)	16	R/W	8000h	28.5.12/ 1055
564	INTC Priority Select Register (INTC_PSR642)	16	R/W	8000h	28.5.12/ 1055
566	INTC Priority Select Register (INTC_PSR643)	16	R/W	8000h	28.5.12/ 1055
568	INTC Priority Select Register (INTC_PSR644)	16	R/W	8000h	28.5.12/ 1055
56A	INTC Priority Select Register (INTC_PSR645)	16	R/W	8000h	28.5.12/ 1055
56C	INTC Priority Select Register (INTC_PSR646)	16	R/W	8000h	28.5.12/ 1055
56E	INTC Priority Select Register (INTC_PSR647)	16	R/W	8000h	28.5.12/ 1055
570	INTC Priority Select Register (INTC_PSR648)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
572	INTC Priority Select Register (INTC_PSR649)	16	R/W	8000h	28.5.12/ 1055
574	INTC Priority Select Register (INTC_PSR650)	16	R/W	8000h	28.5.12/ 1055
576	INTC Priority Select Register (INTC_PSR651)	16	R/W	8000h	28.5.12/ 1055
578	INTC Priority Select Register (INTC_PSR652)	16	R/W	8000h	28.5.12/ 1055
57A	INTC Priority Select Register (INTC_PSR653)	16	R/W	8000h	28.5.12/ 1055
57C	INTC Priority Select Register (INTC_PSR654)	16	R/W	8000h	28.5.12/ 1055
57E	INTC Priority Select Register (INTC_PSR655)	16	R/W	8000h	28.5.12/ 1055
580	INTC Priority Select Register (INTC_PSR656)	16	R/W	8000h	28.5.12/ 1055
582	INTC Priority Select Register (INTC_PSR657)	16	R/W	8000h	28.5.12/ 1055
584	INTC Priority Select Register (INTC_PSR658)	16	R/W	8000h	28.5.12/ 1055
586	INTC Priority Select Register (INTC_PSR659)	16	R/W	8000h	28.5.12/ 1055
588	INTC Priority Select Register (INTC_PSR660)	16	R/W	8000h	28.5.12/ 1055
58A	INTC Priority Select Register (INTC_PSR661)	16	R/W	8000h	28.5.12/ 1055
58C	INTC Priority Select Register (INTC_PSR662)	16	R/W	8000h	28.5.12/ 1055
58E	INTC Priority Select Register (INTC_PSR663)	16	R/W	8000h	28.5.12/ 1055
590	INTC Priority Select Register (INTC_PSR664)	16	R/W	8000h	28.5.12/ 1055
592	INTC Priority Select Register (INTC_PSR665)	16	R/W	8000h	28.5.12/ 1055
594	INTC Priority Select Register (INTC_PSR666)	16	R/W	8000h	28.5.12/ 1055
596	INTC Priority Select Register (INTC_PSR667)	16	R/W	8000h	28.5.12/ 1055
598	INTC Priority Select Register (INTC_PSR668)	16	R/W	8000h	28.5.12/ 1055
59A	INTC Priority Select Register (INTC_PSR669)	16	R/W	8000h	28.5.12/ 1055
59C	INTC Priority Select Register (INTC_PSR670)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
59E	INTC Priority Select Register (INTC_PSR671)	16	R/W	8000h	28.5.12/ 1055
5A0	INTC Priority Select Register (INTC_PSR672)	16	R/W	8000h	28.5.12/ 1055
5A2	INTC Priority Select Register (INTC_PSR673)	16	R/W	8000h	28.5.12/ 1055
5A4	INTC Priority Select Register (INTC_PSR674)	16	R/W	8000h	28.5.12/ 1055
5A6	INTC Priority Select Register (INTC_PSR675)	16	R/W	8000h	28.5.12/ 1055
5A8	INTC Priority Select Register (INTC_PSR676)	16	R/W	8000h	28.5.12/ 1055
5AA	INTC Priority Select Register (INTC_PSR677)	16	R/W	8000h	28.5.12/ 1055
5AC	INTC Priority Select Register (INTC_PSR678)	16	R/W	8000h	28.5.12/ 1055
5AE	INTC Priority Select Register (INTC_PSR679)	16	R/W	8000h	28.5.12/ 1055
5B0	INTC Priority Select Register (INTC_PSR680)	16	R/W	8000h	28.5.12/ 1055
5B2	INTC Priority Select Register (INTC_PSR681)	16	R/W	8000h	28.5.12/ 1055
5B4	INTC Priority Select Register (INTC_PSR682)	16	R/W	8000h	28.5.12/ 1055
5B6	INTC Priority Select Register (INTC_PSR683)	16	R/W	8000h	28.5.12/ 1055
5B8	INTC Priority Select Register (INTC_PSR684)	16	R/W	8000h	28.5.12/ 1055
5BA	INTC Priority Select Register (INTC_PSR685)	16	R/W	8000h	28.5.12/ 1055
5BC	INTC Priority Select Register (INTC_PSR686)	16	R/W	8000h	28.5.12/ 1055
5BE	INTC Priority Select Register (INTC_PSR687)	16	R/W	8000h	28.5.12/ 1055
5C0	INTC Priority Select Register (INTC_PSR688)	16	R/W	8000h	28.5.12/ 1055
5C2	INTC Priority Select Register (INTC_PSR689)	16	R/W	8000h	28.5.12/ 1055
5C4	INTC Priority Select Register (INTC_PSR690)	16	R/W	8000h	28.5.12/ 1055
5C6	INTC Priority Select Register (INTC_PSR691)	16	R/W	8000h	28.5.12/ 1055
5C8	INTC Priority Select Register (INTC_PSR692)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5CA	INTC Priority Select Register (INTC_PSR693)	16	R/W	8000h	28.5.12/ 1055
5CC	INTC Priority Select Register (INTC_PSR694)	16	R/W	8000h	28.5.12/ 1055
5CE	INTC Priority Select Register (INTC_PSR695)	16	R/W	8000h	28.5.12/ 1055
5D0	INTC Priority Select Register (INTC_PSR696)	16	R/W	8000h	28.5.12/ 1055
5D2	INTC Priority Select Register (INTC_PSR697)	16	R/W	8000h	28.5.12/ 1055
5D4	INTC Priority Select Register (INTC_PSR698)	16	R/W	8000h	28.5.12/ 1055
5D6	INTC Priority Select Register (INTC_PSR699)	16	R/W	8000h	28.5.12/ 1055
5D8	INTC Priority Select Register (INTC_PSR700)	16	R/W	8000h	28.5.12/ 1055
5DA	INTC Priority Select Register (INTC_PSR701)	16	R/W	8000h	28.5.12/ 1055
5DC	INTC Priority Select Register (INTC_PSR702)	16	R/W	8000h	28.5.12/ 1055
5DE	INTC Priority Select Register (INTC_PSR703)	16	R/W	8000h	28.5.12/ 1055
5E0	INTC Priority Select Register (INTC_PSR704)	16	R/W	8000h	28.5.12/ 1055
5E2	INTC Priority Select Register (INTC_PSR705)	16	R/W	8000h	28.5.12/ 1055
5E4	INTC Priority Select Register (INTC_PSR706)	16	R/W	8000h	28.5.12/ 1055
5E6	INTC Priority Select Register (INTC_PSR707)	16	R/W	8000h	28.5.12/ 1055
5E8	INTC Priority Select Register (INTC_PSR708)	16	R/W	8000h	28.5.12/ 1055
5EA	INTC Priority Select Register (INTC_PSR709)	16	R/W	8000h	28.5.12/ 1055
5EC	INTC Priority Select Register (INTC_PSR710)	16	R/W	8000h	28.5.12/ 1055
5EE	INTC Priority Select Register (INTC_PSR711)	16	R/W	8000h	28.5.12/ 1055
5F0	INTC Priority Select Register (INTC_PSR712)	16	R/W	8000h	28.5.12/ 1055
5F2	INTC Priority Select Register (INTC_PSR713)	16	R/W	8000h	28.5.12/ 1055
5F4	INTC Priority Select Register (INTC_PSR714)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5F6	INTC Priority Select Register (INTC_PSR715)	16	R/W	8000h	28.5.12/ 1055
5F8	INTC Priority Select Register (INTC_PSR716)	16	R/W	8000h	28.5.12/ 1055
5FA	INTC Priority Select Register (INTC_PSR717)	16	R/W	8000h	28.5.12/ 1055
5FC	INTC Priority Select Register (INTC_PSR718)	16	R/W	8000h	28.5.12/ 1055
5FE	INTC Priority Select Register (INTC_PSR719)	16	R/W	8000h	28.5.12/ 1055
600	INTC Priority Select Register (INTC_PSR720)	16	R/W	8000h	28.5.12/ 1055
602	INTC Priority Select Register (INTC_PSR721)	16	R/W	8000h	28.5.12/ 1055
604	INTC Priority Select Register (INTC_PSR722)	16	R/W	8000h	28.5.12/ 1055
606	INTC Priority Select Register (INTC_PSR723)	16	R/W	8000h	28.5.12/ 1055
608	INTC Priority Select Register (INTC_PSR724)	16	R/W	8000h	28.5.12/ 1055
60A	INTC Priority Select Register (INTC_PSR725)	16	R/W	8000h	28.5.12/ 1055
60C	INTC Priority Select Register (INTC_PSR726)	16	R/W	8000h	28.5.12/ 1055
60E	INTC Priority Select Register (INTC_PSR727)	16	R/W	8000h	28.5.12/ 1055
610	INTC Priority Select Register (INTC_PSR728)	16	R/W	8000h	28.5.12/ 1055
612	INTC Priority Select Register (INTC_PSR729)	16	R/W	8000h	28.5.12/ 1055
614	INTC Priority Select Register (INTC_PSR730)	16	R/W	8000h	28.5.12/ 1055
616	INTC Priority Select Register (INTC_PSR731)	16	R/W	8000h	28.5.12/ 1055
618	INTC Priority Select Register (INTC_PSR732)	16	R/W	8000h	28.5.12/ 1055
61A	INTC Priority Select Register (INTC_PSR733)	16	R/W	8000h	28.5.12/ 1055
61C	INTC Priority Select Register (INTC_PSR734)	16	R/W	8000h	28.5.12/ 1055
61E	INTC Priority Select Register (INTC_PSR735)	16	R/W	8000h	28.5.12/ 1055
620	INTC Priority Select Register (INTC_PSR736)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
622	INTC Priority Select Register (INTC_PSR737)	16	R/W	8000h	28.5.12/ 1055
624	INTC Priority Select Register (INTC_PSR738)	16	R/W	8000h	28.5.12/ 1055
626	INTC Priority Select Register (INTC_PSR739)	16	R/W	8000h	28.5.12/ 1055
628	INTC Priority Select Register (INTC_PSR740)	16	R/W	8000h	28.5.12/ 1055
62A	INTC Priority Select Register (INTC_PSR741)	16	R/W	8000h	28.5.12/ 1055
62C	INTC Priority Select Register (INTC_PSR742)	16	R/W	8000h	28.5.12/ 1055
62E	INTC Priority Select Register (INTC_PSR743)	16	R/W	8000h	28.5.12/ 1055
630	INTC Priority Select Register (INTC_PSR744)	16	R/W	8000h	28.5.12/ 1055
632	INTC Priority Select Register (INTC_PSR745)	16	R/W	8000h	28.5.12/ 1055
634	INTC Priority Select Register (INTC_PSR746)	16	R/W	8000h	28.5.12/ 1055
636	INTC Priority Select Register (INTC_PSR747)	16	R/W	8000h	28.5.12/ 1055
638	INTC Priority Select Register (INTC_PSR748)	16	R/W	8000h	28.5.12/ 1055
63A	INTC Priority Select Register (INTC_PSR749)	16	R/W	8000h	28.5.12/ 1055
63C	INTC Priority Select Register (INTC_PSR750)	16	R/W	8000h	28.5.12/ 1055
63E	INTC Priority Select Register (INTC_PSR751)	16	R/W	8000h	28.5.12/ 1055
640	INTC Priority Select Register (INTC_PSR752)	16	R/W	8000h	28.5.12/ 1055
642	INTC Priority Select Register (INTC_PSR753)	16	R/W	8000h	28.5.12/ 1055
644	INTC Priority Select Register (INTC_PSR754)	16	R/W	8000h	28.5.12/ 1055
646	INTC Priority Select Register (INTC_PSR755)	16	R/W	8000h	28.5.12/ 1055
648	INTC Priority Select Register (INTC_PSR756)	16	R/W	8000h	28.5.12/ 1055
64A	INTC Priority Select Register (INTC_PSR757)	16	R/W	8000h	28.5.12/ 1055
64C	INTC Priority Select Register (INTC_PSR758)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
64E	INTC Priority Select Register (INTC_PSR759)	16	R/W	8000h	28.5.12/ 1055
650	INTC Priority Select Register (INTC_PSR760)	16	R/W	8000h	28.5.12/ 1055
652	INTC Priority Select Register (INTC_PSR761)	16	R/W	8000h	28.5.12/ 1055
654	INTC Priority Select Register (INTC_PSR762)	16	R/W	8000h	28.5.12/ 1055
656	INTC Priority Select Register (INTC_PSR763)	16	R/W	8000h	28.5.12/ 1055
658	INTC Priority Select Register (INTC_PSR764)	16	R/W	8000h	28.5.12/ 1055
65A	INTC Priority Select Register (INTC_PSR765)	16	R/W	8000h	28.5.12/ 1055
65C	INTC Priority Select Register (INTC_PSR766)	16	R/W	8000h	28.5.12/ 1055
65E	INTC Priority Select Register (INTC_PSR767)	16	R/W	8000h	28.5.12/ 1055
660	INTC Priority Select Register (INTC_PSR768)	16	R/W	8000h	28.5.12/ 1055
662	INTC Priority Select Register (INTC_PSR769)	16	R/W	8000h	28.5.12/ 1055
664	INTC Priority Select Register (INTC_PSR770)	16	R/W	8000h	28.5.12/ 1055
666	INTC Priority Select Register (INTC_PSR771)	16	R/W	8000h	28.5.12/ 1055
668	INTC Priority Select Register (INTC_PSR772)	16	R/W	8000h	28.5.12/ 1055
66A	INTC Priority Select Register (INTC_PSR773)	16	R/W	8000h	28.5.12/ 1055
66C	INTC Priority Select Register (INTC_PSR774)	16	R/W	8000h	28.5.12/ 1055
66E	INTC Priority Select Register (INTC_PSR775)	16	R/W	8000h	28.5.12/ 1055
670	INTC Priority Select Register (INTC_PSR776)	16	R/W	8000h	28.5.12/ 1055
672	INTC Priority Select Register (INTC_PSR777)	16	R/W	8000h	28.5.12/ 1055
674	INTC Priority Select Register (INTC_PSR778)	16	R/W	8000h	28.5.12/ 1055
676	INTC Priority Select Register (INTC_PSR779)	16	R/W	8000h	28.5.12/ 1055
678	INTC Priority Select Register (INTC_PSR780)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
67A	INTC Priority Select Register (INTC_PSR781)	16	R/W	8000h	28.5.12/ 1055
67C	INTC Priority Select Register (INTC_PSR782)	16	R/W	8000h	28.5.12/ 1055
67E	INTC Priority Select Register (INTC_PSR783)	16	R/W	8000h	28.5.12/ 1055
680	INTC Priority Select Register (INTC_PSR784)	16	R/W	8000h	28.5.12/ 1055
682	INTC Priority Select Register (INTC_PSR785)	16	R/W	8000h	28.5.12/ 1055
684	INTC Priority Select Register (INTC_PSR786)	16	R/W	8000h	28.5.12/ 1055
686	INTC Priority Select Register (INTC_PSR787)	16	R/W	8000h	28.5.12/ 1055
688	INTC Priority Select Register (INTC_PSR788)	16	R/W	8000h	28.5.12/ 1055
68A	INTC Priority Select Register (INTC_PSR789)	16	R/W	8000h	28.5.12/ 1055
68C	INTC Priority Select Register (INTC_PSR790)	16	R/W	8000h	28.5.12/ 1055
68E	INTC Priority Select Register (INTC_PSR791)	16	R/W	8000h	28.5.12/ 1055
690	INTC Priority Select Register (INTC_PSR792)	16	R/W	8000h	28.5.12/ 1055
692	INTC Priority Select Register (INTC_PSR793)	16	R/W	8000h	28.5.12/ 1055
694	INTC Priority Select Register (INTC_PSR794)	16	R/W	8000h	28.5.12/ 1055
696	INTC Priority Select Register (INTC_PSR795)	16	R/W	8000h	28.5.12/ 1055
698	INTC Priority Select Register (INTC_PSR796)	16	R/W	8000h	28.5.12/ 1055
69A	INTC Priority Select Register (INTC_PSR797)	16	R/W	8000h	28.5.12/ 1055
69C	INTC Priority Select Register (INTC_PSR798)	16	R/W	8000h	28.5.12/ 1055
69E	INTC Priority Select Register (INTC_PSR799)	16	R/W	8000h	28.5.12/ 1055
6A0	INTC Priority Select Register (INTC_PSR800)	16	R/W	8000h	28.5.12/ 1055
6A2	INTC Priority Select Register (INTC_PSR801)	16	R/W	8000h	28.5.12/ 1055
6A4	INTC Priority Select Register (INTC_PSR802)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6A6	INTC Priority Select Register (INTC_PSR803)	16	R/W	8000h	28.5.12/ 1055
6A8	INTC Priority Select Register (INTC_PSR804)	16	R/W	8000h	28.5.12/ 1055
6AA	INTC Priority Select Register (INTC_PSR805)	16	R/W	8000h	28.5.12/ 1055
6AC	INTC Priority Select Register (INTC_PSR806)	16	R/W	8000h	28.5.12/ 1055
6AE	INTC Priority Select Register (INTC_PSR807)	16	R/W	8000h	28.5.12/ 1055
6B0	INTC Priority Select Register (INTC_PSR808)	16	R/W	8000h	28.5.12/ 1055
6B2	INTC Priority Select Register (INTC_PSR809)	16	R/W	8000h	28.5.12/ 1055
6B4	INTC Priority Select Register (INTC_PSR810)	16	R/W	8000h	28.5.12/ 1055
6B6	INTC Priority Select Register (INTC_PSR811)	16	R/W	8000h	28.5.12/ 1055
6B8	INTC Priority Select Register (INTC_PSR812)	16	R/W	8000h	28.5.12/ 1055
6BA	INTC Priority Select Register (INTC_PSR813)	16	R/W	8000h	28.5.12/ 1055
6BC	INTC Priority Select Register (INTC_PSR814)	16	R/W	8000h	28.5.12/ 1055
6BE	INTC Priority Select Register (INTC_PSR815)	16	R/W	8000h	28.5.12/ 1055
6C0	INTC Priority Select Register (INTC_PSR816)	16	R/W	8000h	28.5.12/ 1055
6C2	INTC Priority Select Register (INTC_PSR817)	16	R/W	8000h	28.5.12/ 1055
6C4	INTC Priority Select Register (INTC_PSR818)	16	R/W	8000h	28.5.12/ 1055
6C6	INTC Priority Select Register (INTC_PSR819)	16	R/W	8000h	28.5.12/ 1055
6C8	INTC Priority Select Register (INTC_PSR820)	16	R/W	8000h	28.5.12/ 1055
6CA	INTC Priority Select Register (INTC_PSR821)	16	R/W	8000h	28.5.12/ 1055
6CC	INTC Priority Select Register (INTC_PSR822)	16	R/W	8000h	28.5.12/ 1055
6CE	INTC Priority Select Register (INTC_PSR823)	16	R/W	8000h	28.5.12/ 1055
6D0	INTC Priority Select Register (INTC_PSR824)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6D2	INTC Priority Select Register (INTC_PSR825)	16	R/W	8000h	28.5.12/ 1055
6D4	INTC Priority Select Register (INTC_PSR826)	16	R/W	8000h	28.5.12/ 1055
6D6	INTC Priority Select Register (INTC_PSR827)	16	R/W	8000h	28.5.12/ 1055
6D8	INTC Priority Select Register (INTC_PSR828)	16	R/W	8000h	28.5.12/ 1055
6DA	INTC Priority Select Register (INTC_PSR829)	16	R/W	8000h	28.5.12/ 1055
6DC	INTC Priority Select Register (INTC_PSR830)	16	R/W	8000h	28.5.12/ 1055
6DE	INTC Priority Select Register (INTC_PSR831)	16	R/W	8000h	28.5.12/ 1055
6E0	INTC Priority Select Register (INTC_PSR832)	16	R/W	8000h	28.5.12/ 1055
6E2	INTC Priority Select Register (INTC_PSR833)	16	R/W	8000h	28.5.12/ 1055
6E4	INTC Priority Select Register (INTC_PSR834)	16	R/W	8000h	28.5.12/ 1055
6E6	INTC Priority Select Register (INTC_PSR835)	16	R/W	8000h	28.5.12/ 1055
6E8	INTC Priority Select Register (INTC_PSR836)	16	R/W	8000h	28.5.12/ 1055
6EA	INTC Priority Select Register (INTC_PSR837)	16	R/W	8000h	28.5.12/ 1055
6EC	INTC Priority Select Register (INTC_PSR838)	16	R/W	8000h	28.5.12/ 1055
6EE	INTC Priority Select Register (INTC_PSR839)	16	R/W	8000h	28.5.12/ 1055
6F0	INTC Priority Select Register (INTC_PSR840)	16	R/W	8000h	28.5.12/ 1055
6F2	INTC Priority Select Register (INTC_PSR841)	16	R/W	8000h	28.5.12/ 1055
6F4	INTC Priority Select Register (INTC_PSR842)	16	R/W	8000h	28.5.12/ 1055
6F6	INTC Priority Select Register (INTC_PSR843)	16	R/W	8000h	28.5.12/ 1055
6F8	INTC Priority Select Register (INTC_PSR844)	16	R/W	8000h	28.5.12/ 1055
6FA	INTC Priority Select Register (INTC_PSR845)	16	R/W	8000h	28.5.12/ 1055
6FC	INTC Priority Select Register (INTC_PSR846)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6FE	INTC Priority Select Register (INTC_PSR847)	16	R/W	8000h	28.5.12/ 1055
700	INTC Priority Select Register (INTC_PSR848)	16	R/W	8000h	28.5.12/ 1055
702	INTC Priority Select Register (INTC_PSR849)	16	R/W	8000h	28.5.12/ 1055
704	INTC Priority Select Register (INTC_PSR850)	16	R/W	8000h	28.5.12/ 1055
706	INTC Priority Select Register (INTC_PSR851)	16	R/W	8000h	28.5.12/ 1055
708	INTC Priority Select Register (INTC_PSR852)	16	R/W	8000h	28.5.12/ 1055
70A	INTC Priority Select Register (INTC_PSR853)	16	R/W	8000h	28.5.12/ 1055
70C	INTC Priority Select Register (INTC_PSR854)	16	R/W	8000h	28.5.12/ 1055
70E	INTC Priority Select Register (INTC_PSR855)	16	R/W	8000h	28.5.12/ 1055
710	INTC Priority Select Register (INTC_PSR856)	16	R/W	8000h	28.5.12/ 1055
712	INTC Priority Select Register (INTC_PSR857)	16	R/W	8000h	28.5.12/ 1055
714	INTC Priority Select Register (INTC_PSR858)	16	R/W	8000h	28.5.12/ 1055
716	INTC Priority Select Register (INTC_PSR859)	16	R/W	8000h	28.5.12/ 1055
718	INTC Priority Select Register (INTC_PSR860)	16	R/W	8000h	28.5.12/ 1055
71A	INTC Priority Select Register (INTC_PSR861)	16	R/W	8000h	28.5.12/ 1055
71C	INTC Priority Select Register (INTC_PSR862)	16	R/W	8000h	28.5.12/ 1055
71E	INTC Priority Select Register (INTC_PSR863)	16	R/W	8000h	28.5.12/ 1055
720	INTC Priority Select Register (INTC_PSR864)	16	R/W	8000h	28.5.12/ 1055
722	INTC Priority Select Register (INTC_PSR865)	16	R/W	8000h	28.5.12/ 1055
724	INTC Priority Select Register (INTC_PSR866)	16	R/W	8000h	28.5.12/ 1055
726	INTC Priority Select Register (INTC_PSR867)	16	R/W	8000h	28.5.12/ 1055
728	INTC Priority Select Register (INTC_PSR868)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
72A	INTC Priority Select Register (INTC_PSR869)	16	R/W	8000h	28.5.12/ 1055
72C	INTC Priority Select Register (INTC_PSR870)	16	R/W	8000h	28.5.12/ 1055
72E	INTC Priority Select Register (INTC_PSR871)	16	R/W	8000h	28.5.12/ 1055
730	INTC Priority Select Register (INTC_PSR872)	16	R/W	8000h	28.5.12/ 1055
732	INTC Priority Select Register (INTC_PSR873)	16	R/W	8000h	28.5.12/ 1055
734	INTC Priority Select Register (INTC_PSR874)	16	R/W	8000h	28.5.12/ 1055
736	INTC Priority Select Register (INTC_PSR875)	16	R/W	8000h	28.5.12/ 1055
738	INTC Priority Select Register (INTC_PSR876)	16	R/W	8000h	28.5.12/ 1055
73A	INTC Priority Select Register (INTC_PSR877)	16	R/W	8000h	28.5.12/ 1055
73C	INTC Priority Select Register (INTC_PSR878)	16	R/W	8000h	28.5.12/ 1055
73E	INTC Priority Select Register (INTC_PSR879)	16	R/W	8000h	28.5.12/ 1055
740	INTC Priority Select Register (INTC_PSR880)	16	R/W	8000h	28.5.12/ 1055
742	INTC Priority Select Register (INTC_PSR881)	16	R/W	8000h	28.5.12/ 1055
744	INTC Priority Select Register (INTC_PSR882)	16	R/W	8000h	28.5.12/ 1055
746	INTC Priority Select Register (INTC_PSR883)	16	R/W	8000h	28.5.12/ 1055
748	INTC Priority Select Register (INTC_PSR884)	16	R/W	8000h	28.5.12/ 1055
74A	INTC Priority Select Register (INTC_PSR885)	16	R/W	8000h	28.5.12/ 1055
74C	INTC Priority Select Register (INTC_PSR886)	16	R/W	8000h	28.5.12/ 1055
74E	INTC Priority Select Register (INTC_PSR887)	16	R/W	8000h	28.5.12/ 1055
750	INTC Priority Select Register (INTC_PSR888)	16	R/W	8000h	28.5.12/ 1055
752	INTC Priority Select Register (INTC_PSR889)	16	R/W	8000h	28.5.12/ 1055
754	INTC Priority Select Register (INTC_PSR890)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
756	INTC Priority Select Register (INTC_PSR891)	16	R/W	8000h	28.5.12/ 1055
758	INTC Priority Select Register (INTC_PSR892)	16	R/W	8000h	28.5.12/ 1055
75A	INTC Priority Select Register (INTC_PSR893)	16	R/W	8000h	28.5.12/ 1055
75C	INTC Priority Select Register (INTC_PSR894)	16	R/W	8000h	28.5.12/ 1055
75E	INTC Priority Select Register (INTC_PSR895)	16	R/W	8000h	28.5.12/ 1055
760	INTC Priority Select Register (INTC_PSR896)	16	R/W	8000h	28.5.12/ 1055
762	INTC Priority Select Register (INTC_PSR897)	16	R/W	8000h	28.5.12/ 1055
764	INTC Priority Select Register (INTC_PSR898)	16	R/W	8000h	28.5.12/ 1055
766	INTC Priority Select Register (INTC_PSR899)	16	R/W	8000h	28.5.12/ 1055
768	INTC Priority Select Register (INTC_PSR900)	16	R/W	8000h	28.5.12/ 1055
76A	INTC Priority Select Register (INTC_PSR901)	16	R/W	8000h	28.5.12/ 1055
76C	INTC Priority Select Register (INTC_PSR902)	16	R/W	8000h	28.5.12/ 1055
76E	INTC Priority Select Register (INTC_PSR903)	16	R/W	8000h	28.5.12/ 1055
770	INTC Priority Select Register (INTC_PSR904)	16	R/W	8000h	28.5.12/ 1055
772	INTC Priority Select Register (INTC_PSR905)	16	R/W	8000h	28.5.12/ 1055
774	INTC Priority Select Register (INTC_PSR906)	16	R/W	8000h	28.5.12/ 1055
776	INTC Priority Select Register (INTC_PSR907)	16	R/W	8000h	28.5.12/ 1055
778	INTC Priority Select Register (INTC_PSR908)	16	R/W	8000h	28.5.12/ 1055
77A	INTC Priority Select Register (INTC_PSR909)	16	R/W	8000h	28.5.12/ 1055
77C	INTC Priority Select Register (INTC_PSR910)	16	R/W	8000h	28.5.12/ 1055
77E	INTC Priority Select Register (INTC_PSR911)	16	R/W	8000h	28.5.12/ 1055
780	INTC Priority Select Register (INTC_PSR912)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
782	INTC Priority Select Register (INTC_PSR913)	16	R/W	8000h	28.5.12/ 1055
784	INTC Priority Select Register (INTC_PSR914)	16	R/W	8000h	28.5.12/ 1055
786	INTC Priority Select Register (INTC_PSR915)	16	R/W	8000h	28.5.12/ 1055
788	INTC Priority Select Register (INTC_PSR916)	16	R/W	8000h	28.5.12/ 1055
78A	INTC Priority Select Register (INTC_PSR917)	16	R/W	8000h	28.5.12/ 1055
78C	INTC Priority Select Register (INTC_PSR918)	16	R/W	8000h	28.5.12/ 1055
78E	INTC Priority Select Register (INTC_PSR919)	16	R/W	8000h	28.5.12/ 1055
790	INTC Priority Select Register (INTC_PSR920)	16	R/W	8000h	28.5.12/ 1055
792	INTC Priority Select Register (INTC_PSR921)	16	R/W	8000h	28.5.12/ 1055
794	INTC Priority Select Register (INTC_PSR922)	16	R/W	8000h	28.5.12/ 1055
796	INTC Priority Select Register (INTC_PSR923)	16	R/W	8000h	28.5.12/ 1055
798	INTC Priority Select Register (INTC_PSR924)	16	R/W	8000h	28.5.12/ 1055
79A	INTC Priority Select Register (INTC_PSR925)	16	R/W	8000h	28.5.12/ 1055
79C	INTC Priority Select Register (INTC_PSR926)	16	R/W	8000h	28.5.12/ 1055
79E	INTC Priority Select Register (INTC_PSR927)	16	R/W	8000h	28.5.12/ 1055
7A0	INTC Priority Select Register (INTC_PSR928)	16	R/W	8000h	28.5.12/ 1055
7A2	INTC Priority Select Register (INTC_PSR929)	16	R/W	8000h	28.5.12/ 1055
7A4	INTC Priority Select Register (INTC_PSR930)	16	R/W	8000h	28.5.12/ 1055
7A6	INTC Priority Select Register (INTC_PSR931)	16	R/W	8000h	28.5.12/ 1055
7A8	INTC Priority Select Register (INTC_PSR932)	16	R/W	8000h	28.5.12/ 1055
7AA	INTC Priority Select Register (INTC_PSR933)	16	R/W	8000h	28.5.12/ 1055
7AC	INTC Priority Select Register (INTC_PSR934)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7AE	INTC Priority Select Register (INTC_PSR935)	16	R/W	8000h	28.5.12/ 1055
7B0	INTC Priority Select Register (INTC_PSR936)	16	R/W	8000h	28.5.12/ 1055
7B2	INTC Priority Select Register (INTC_PSR937)	16	R/W	8000h	28.5.12/ 1055
7B4	INTC Priority Select Register (INTC_PSR938)	16	R/W	8000h	28.5.12/ 1055
7B6	INTC Priority Select Register (INTC_PSR939)	16	R/W	8000h	28.5.12/ 1055
7B8	INTC Priority Select Register (INTC_PSR940)	16	R/W	8000h	28.5.12/ 1055
7BA	INTC Priority Select Register (INTC_PSR941)	16	R/W	8000h	28.5.12/ 1055
7BC	INTC Priority Select Register (INTC_PSR942)	16	R/W	8000h	28.5.12/ 1055
7BE	INTC Priority Select Register (INTC_PSR943)	16	R/W	8000h	28.5.12/ 1055
7C0	INTC Priority Select Register (INTC_PSR944)	16	R/W	8000h	28.5.12/ 1055
7C2	INTC Priority Select Register (INTC_PSR945)	16	R/W	8000h	28.5.12/ 1055
7C4	INTC Priority Select Register (INTC_PSR946)	16	R/W	8000h	28.5.12/ 1055
7C6	INTC Priority Select Register (INTC_PSR947)	16	R/W	8000h	28.5.12/ 1055
7C8	INTC Priority Select Register (INTC_PSR948)	16	R/W	8000h	28.5.12/ 1055
7CA	INTC Priority Select Register (INTC_PSR949)	16	R/W	8000h	28.5.12/ 1055
7CC	INTC Priority Select Register (INTC_PSR950)	16	R/W	8000h	28.5.12/ 1055
7CE	INTC Priority Select Register (INTC_PSR951)	16	R/W	8000h	28.5.12/ 1055
7D0	INTC Priority Select Register (INTC_PSR952)	16	R/W	8000h	28.5.12/ 1055
7D2	INTC Priority Select Register (INTC_PSR953)	16	R/W	8000h	28.5.12/ 1055
7D4	INTC Priority Select Register (INTC_PSR954)	16	R/W	8000h	28.5.12/ 1055
7D6	INTC Priority Select Register (INTC_PSR955)	16	R/W	8000h	28.5.12/ 1055
7D8	INTC Priority Select Register (INTC_PSR956)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7DA	INTC Priority Select Register (INTC_PSR957)	16	R/W	8000h	28.5.12/ 1055
7DC	INTC Priority Select Register (INTC_PSR958)	16	R/W	8000h	28.5.12/ 1055
7DE	INTC Priority Select Register (INTC_PSR959)	16	R/W	8000h	28.5.12/ 1055
7E0	INTC Priority Select Register (INTC_PSR960)	16	R/W	8000h	28.5.12/ 1055
7E2	INTC Priority Select Register (INTC_PSR961)	16	R/W	8000h	28.5.12/ 1055
7E4	INTC Priority Select Register (INTC_PSR962)	16	R/W	8000h	28.5.12/ 1055
7E6	INTC Priority Select Register (INTC_PSR963)	16	R/W	8000h	28.5.12/ 1055
7E8	INTC Priority Select Register (INTC_PSR964)	16	R/W	8000h	28.5.12/ 1055
7EA	INTC Priority Select Register (INTC_PSR965)	16	R/W	8000h	28.5.12/ 1055
7EC	INTC Priority Select Register (INTC_PSR966)	16	R/W	8000h	28.5.12/ 1055
7EE	INTC Priority Select Register (INTC_PSR967)	16	R/W	8000h	28.5.12/ 1055
7F0	INTC Priority Select Register (INTC_PSR968)	16	R/W	8000h	28.5.12/ 1055
7F2	INTC Priority Select Register (INTC_PSR969)	16	R/W	8000h	28.5.12/ 1055
7F4	INTC Priority Select Register (INTC_PSR970)	16	R/W	8000h	28.5.12/ 1055
7F6	INTC Priority Select Register (INTC_PSR971)	16	R/W	8000h	28.5.12/ 1055
7F8	INTC Priority Select Register (INTC_PSR972)	16	R/W	8000h	28.5.12/ 1055
7FA	INTC Priority Select Register (INTC_PSR973)	16	R/W	8000h	28.5.12/ 1055
7FC	INTC Priority Select Register (INTC_PSR974)	16	R/W	8000h	28.5.12/ 1055
7FE	INTC Priority Select Register (INTC_PSR975)	16	R/W	8000h	28.5.12/ 1055
800	INTC Priority Select Register (INTC_PSR976)	16	R/W	8000h	28.5.12/ 1055
802	INTC Priority Select Register (INTC_PSR977)	16	R/W	8000h	28.5.12/ 1055
804	INTC Priority Select Register (INTC_PSR978)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
806	INTC Priority Select Register (INTC_PSR979)	16	R/W	8000h	28.5.12/ 1055
808	INTC Priority Select Register (INTC_PSR980)	16	R/W	8000h	28.5.12/ 1055
80A	INTC Priority Select Register (INTC_PSR981)	16	R/W	8000h	28.5.12/ 1055
80C	INTC Priority Select Register (INTC_PSR982)	16	R/W	8000h	28.5.12/ 1055
80E	INTC Priority Select Register (INTC_PSR983)	16	R/W	8000h	28.5.12/ 1055
810	INTC Priority Select Register (INTC_PSR984)	16	R/W	8000h	28.5.12/ 1055
812	INTC Priority Select Register (INTC_PSR985)	16	R/W	8000h	28.5.12/ 1055
814	INTC Priority Select Register (INTC_PSR986)	16	R/W	8000h	28.5.12/ 1055
816	INTC Priority Select Register (INTC_PSR987)	16	R/W	8000h	28.5.12/ 1055
818	INTC Priority Select Register (INTC_PSR988)	16	R/W	8000h	28.5.12/ 1055
81A	INTC Priority Select Register (INTC_PSR989)	16	R/W	8000h	28.5.12/ 1055
81C	INTC Priority Select Register (INTC_PSR990)	16	R/W	8000h	28.5.12/ 1055
81E	INTC Priority Select Register (INTC_PSR991)	16	R/W	8000h	28.5.12/ 1055
820	INTC Priority Select Register (INTC_PSR992)	16	R/W	8000h	28.5.12/ 1055
822	INTC Priority Select Register (INTC_PSR993)	16	R/W	8000h	28.5.12/ 1055
824	INTC Priority Select Register (INTC_PSR994)	16	R/W	8000h	28.5.12/ 1055
826	INTC Priority Select Register (INTC_PSR995)	16	R/W	8000h	28.5.12/ 1055
828	INTC Priority Select Register (INTC_PSR996)	16	R/W	8000h	28.5.12/ 1055
82A	INTC Priority Select Register (INTC_PSR997)	16	R/W	8000h	28.5.12/ 1055
82C	INTC Priority Select Register (INTC_PSR998)	16	R/W	8000h	28.5.12/ 1055
82E	INTC Priority Select Register (INTC_PSR999)	16	R/W	8000h	28.5.12/ 1055
830	INTC Priority Select Register (INTC_PSR1000)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
832	INTC Priority Select Register (INTC_PSR1001)	16	R/W	8000h	28.5.12/ 1055
834	INTC Priority Select Register (INTC_PSR1002)	16	R/W	8000h	28.5.12/ 1055
836	INTC Priority Select Register (INTC_PSR1003)	16	R/W	8000h	28.5.12/ 1055
838	INTC Priority Select Register (INTC_PSR1004)	16	R/W	8000h	28.5.12/ 1055
83A	INTC Priority Select Register (INTC_PSR1005)	16	R/W	8000h	28.5.12/ 1055
83C	INTC Priority Select Register (INTC_PSR1006)	16	R/W	8000h	28.5.12/ 1055
83E	INTC Priority Select Register (INTC_PSR1007)	16	R/W	8000h	28.5.12/ 1055
840	INTC Priority Select Register (INTC_PSR1008)	16	R/W	8000h	28.5.12/ 1055
842	INTC Priority Select Register (INTC_PSR1009)	16	R/W	8000h	28.5.12/ 1055
844	INTC Priority Select Register (INTC_PSR1010)	16	R/W	8000h	28.5.12/ 1055
846	INTC Priority Select Register (INTC_PSR1011)	16	R/W	8000h	28.5.12/ 1055
848	INTC Priority Select Register (INTC_PSR1012)	16	R/W	8000h	28.5.12/ 1055
84A	INTC Priority Select Register (INTC_PSR1013)	16	R/W	8000h	28.5.12/ 1055
84C	INTC Priority Select Register (INTC_PSR1014)	16	R/W	8000h	28.5.12/ 1055
84E	INTC Priority Select Register (INTC_PSR1015)	16	R/W	8000h	28.5.12/ 1055
850	INTC Priority Select Register (INTC_PSR1016)	16	R/W	8000h	28.5.12/ 1055
852	INTC Priority Select Register (INTC_PSR1017)	16	R/W	8000h	28.5.12/ 1055
854	INTC Priority Select Register (INTC_PSR1018)	16	R/W	8000h	28.5.12/ 1055
856	INTC Priority Select Register (INTC_PSR1019)	16	R/W	8000h	28.5.12/ 1055
858	INTC Priority Select Register (INTC_PSR1020)	16	R/W	8000h	28.5.12/ 1055
85A	INTC Priority Select Register (INTC_PSR1021)	16	R/W	8000h	28.5.12/ 1055
85C	INTC Priority Select Register (INTC_PSR1022)	16	R/W	8000h	28.5.12/ 1055

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
85E	INTC Priority Select Register (INTC_PSR1023)	16	R/W	8000h	28.5.12/ 1055
1000	INTC Monitor Mode Register 0 (INTC_MMRC0)	32	R/W	0000_0000h	28.5.13/ 1056
1004	INTC HIPRI Register (INTC_HIPRI0C0)	32	R/W	0000_0000h	28.5.14/ 1057
1008	INTC HIPRI Register (INTC_HIPRI1C0)	32	R/W	0000_0000h	28.5.14/ 1057
100C	INTC HIPRI Register (INTC_HIPRI2C0)	32	R/W	0000_0000h	28.5.14/ 1057
1014	INTC LAT Register (INTC_LAT0C0)	32	R/W	0000_0000h	28.5.15/ 1057
1018	INTC LAT Register (INTC_LAT1C0)	32	R/W	0000_0000h	28.5.15/ 1057
101C	INTC LAT Register (INTC_LAT2C0)	32	R/W	0000_0000h	28.5.15/ 1057
1024	INTC Timer Register (INTC_TIMER0C0)	32	R/W	0000_0000h	28.5.16/ 1058
1028	INTC Timer Register (INTC_TIMER1C0)	32	R/W	0000_0000h	28.5.16/ 1058
102C	INTC Timer Register (INTC_TIMER2C0)	32	R/W	0000_0000h	28.5.16/ 1058
1034	INTC Monitor Mode Register 1 (INTC_MMRC1)	32	R/W	0000_0000h	28.5.17/ 1058
1038	INTC HIPRI Register (INTC_HIPRI0C1)	32	R/W	0000_0000h	28.5.18/ 1059
103C	INTC HIPRI Register (INTC_HIPRI1C1)	32	R/W	0000_0000h	28.5.18/ 1059
1040	INTC HIPRI Register (INTC_HIPRI2C1)	32	R/W	0000_0000h	28.5.18/ 1059
1048	INTC LAT Register (INTC_LAT0C1)	32	R/W	0000_0000h	28.5.19/ 1060
104C	INTC LAT Register (INTC_LAT1C1)	32	R/W	0000_0000h	28.5.19/ 1060
1050	INTC LAT Register (INTC_LAT2C1)	32	R/W	0000_0000h	28.5.19/ 1060
1058	INTC Timer Register (INTC_TIMER0C1)	32	R/W	0000_0000h	28.5.20/ 1060
105C	INTC Timer Register (INTC_TIMER1C1)	32	R/W	0000_0000h	28.5.20/ 1060
1060	INTC Timer Register (INTC_TIMER2C1)	32	R/W	0000_0000h	28.5.20/ 1060
1068	INTC Monitor Mode Register 2 (INTC_MMRC2)	32	R/W	0000_0000h	28.5.21/ 1061

Table continues on the next page...

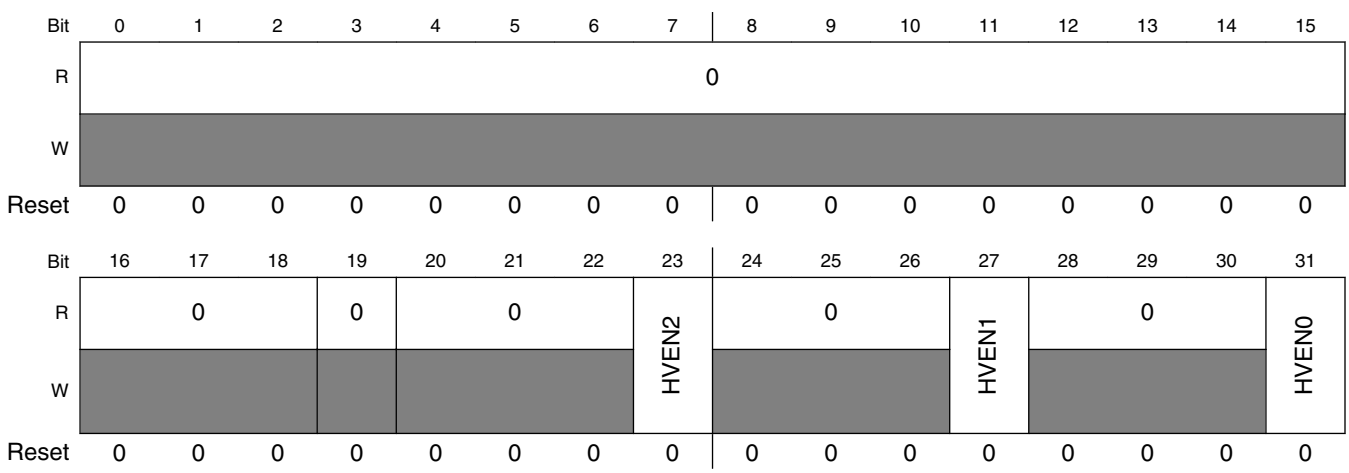
INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
106C	INTC HIPRI Register (INTC_HIPRI0C2)	32	R/W	0000_0000h	28.5.22/ 1061
1070	INTC HIPRI Register (INTC_HIPRI1C2)	32	R/W	0000_0000h	28.5.22/ 1061
1074	INTC HIPRI Register (INTC_HIPRI2C2)	32	R/W	0000_0000h	28.5.22/ 1061
107C	INTC LAT Register (INTC_LAT0C2)	32	R/W	0000_0000h	28.5.23/ 1062
1080	INTC LAT Register (INTC_LAT1C2)	32	R/W	0000_0000h	28.5.23/ 1062
1084	INTC LAT Register (INTC_LAT2C2)	32	R/W	0000_0000h	28.5.23/ 1062
108C	INTC Timer Register (INTC_TIMER0C2)	32	R/W	0000_0000h	28.5.24/ 1062
1090	INTC Timer Register (INTC_TIMER1C2)	32	R/W	0000_0000h	28.5.24/ 1062
1094	INTC Timer Register (INTC_TIMER2C2)	32	R/W	0000_0000h	28.5.24/ 1062

28.5.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

Address: 0h base + 0h offset = 0h



INTC_BCR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HVEN2	Hardware vector enable for processor 2. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 HVEN1	Hardware vector enable for processor 1. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN0	Hardware vector enable for processor 0. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

28.5.2 INTC Current Priority Register for Processor 0 (INTC_CPR0)

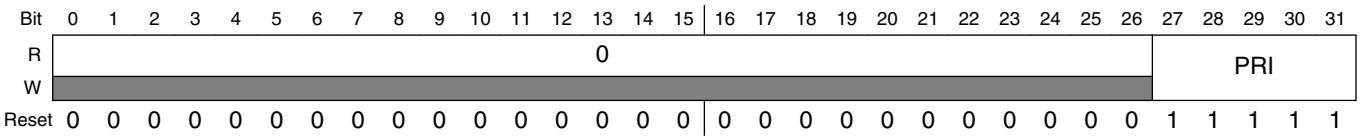
The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 10h offset = 10h



INTC_CPR0 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PRI	Priority of the currently executing ISR, according to the field values 31 (highest priority) down to 0 (lowest priority).

28.5.3 INTC Current Priority Register for Processor 1 (INTC_CPR1)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 14h offset = 14h

Bit	0–26	27–31
R	0	PRI
W		
Reset	0	1 1 1 1 1

INTC_CPR1 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PRI	Priority of the currently executing ISR, according to the field values 31 (highest priority) down to 0 (lowest priority).

28.5.4 INTC Current Priority Register for Processor 2 (INTC_CPR2)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 18h offset = 18h

Bit	0–26	27–31
R	0	PRI
W		
Reset	0	1 1 1 1 1

INTC_CPR2 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PRI	Priority of the currently executing ISR, according to the field values 31 (highest priority) down to 0 (lowest priority).

28.5.5 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 20h offset = 20h

Bit	0–15	16–19	20–29	30–31
R	VTBA		INTVEC	0
W				
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0

INTC_IACKR0 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.5.6 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA																				INTVEC										0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IACKR1 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.5.7 INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA																				INTVEC										0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IACKR2 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.5.8 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	EOI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_EOIR0 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

28.5.9 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	EOI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_EOIR1 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

28.5.10 INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Address: 0h base + 38h offset = 38h

Bit	0–15	16–31
R		
W	EOI	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

INTC_EOIR2 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

28.5.11 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

See Figure 28-2 for limitations on writing to this register.

Address: 0h base + 40h offset + (1d × i), where i=0d to 15d

Bit	0–3	4	5	6	7
Read	0	0		0	CLR
Write				SET	w1c
Reset	0	0	0	0	0

INTC_SSCIR*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bits. CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

28.5.12 INTC Priority Select Register (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32-1023, and their priorities are configured in INTC_PSR32-1023, respectively.

See Figure 28-2 for limitations on writing to this register.

NOTE

Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

NOTE

Interrupts are not detected while the PSRs are being written.

Address: 0h base + 60h offset + (2d × i), where i=0d to 1023d

Bit	0	1	2	3	4	5	6	7
Read	PRC_SELN0	PRC_SELN1	PRC_SELN2	0	0			SWTN
Write								
Reset	1	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			PRIN				
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PSR*n* field descriptions

Field	Description
0 PRC_SELN0	Processor select 0. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 0. 0 Interrupt request not sent to processor 0 1 Interrupt request sent to processor 0
1 PRC_SELN1	Processor select 1. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 1.

Table continues on the next page...

INTC_PSR*n* field descriptions (continued)

Field	Description
	0 Interrupt request not sent to processor 1 1 Interrupt request sent to processor 1
2 PRC_SELN2	Processor select 2. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 2. 0 Interrupt request not sent to processor 2 1 Interrupt request sent to processor 2
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWTN	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PRIN	Priority select. Selects the priority for the interrupt requests. PRI has values 31 (highest priority) down to 0 (lowest priority).

28.5.13 INTC Monitor Mode Register 0 (INTC_MMRC0)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1000h offset = 1000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														MM	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_MMRC0 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero.

Table continues on the next page...

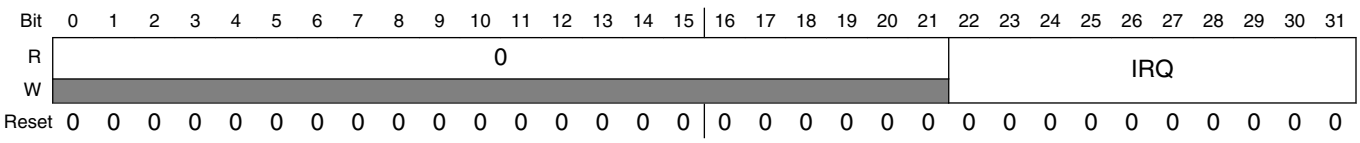
INTC_MMRC0 field descriptions (continued)

Field	Description
	10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

28.5.14 INTC HIPRI Register (INTC_HIPRI*n*C0)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1004h offset + (4d × i), where i=0d to 2d



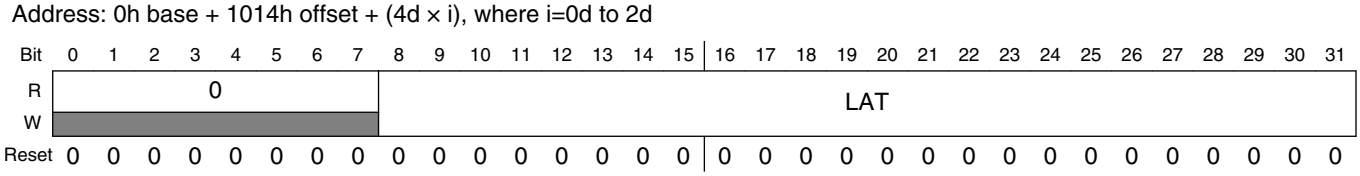
INTC_HIPRI*n*C0 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

28.5.15 INTC LAT Register (INTC_LAT*n*C0)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1014h offset + (4d × i), where i=0d to 2d



INTC_LAT*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

28.5.16 INTC Timer Register (INTC_TIMER*n*C0)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the 'm' is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 1024h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TIMER																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_TIMER*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

28.5.17 INTC Monitor Mode Register 1 (INTC_MMRC1)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1034h offset = 1034h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														MM	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_MMRC1 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

28.5.18 INTC HIPRI Register (INTC_HIPRI*n*C1)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1038h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						IRQ									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_HIPRI*n*C1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

28.5.19 INTC LAT Register (INTC_LAT*n*C1)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1048h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LAT																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_LAT*n*C1 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

28.5.20 INTC Timer Register (INTC_TIMER*n*C1)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 1058h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TIMER																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

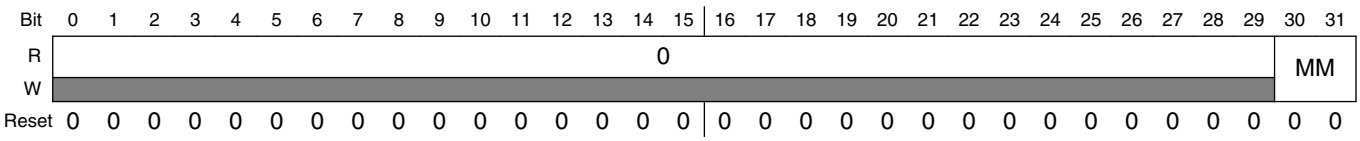
INTC_TIMER*n*C1 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

28.5.21 INTC Monitor Mode Register 2 (INTC_MMRC2)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1068h offset = 1068h



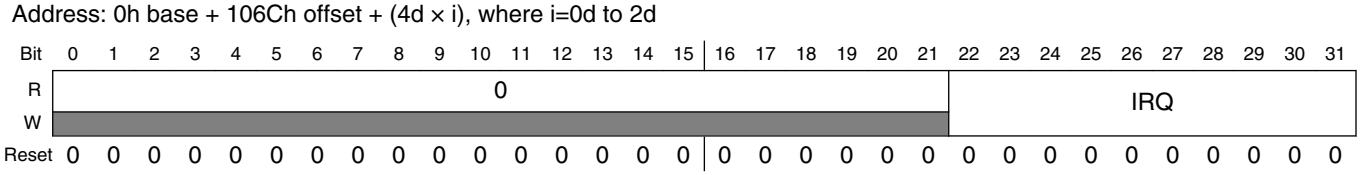
INTC_MMRC2 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

28.5.22 INTC HIPRI Register (INTC_HIPRI*n*C2)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 106Ch offset + (4d × i), where i=0d to 2d



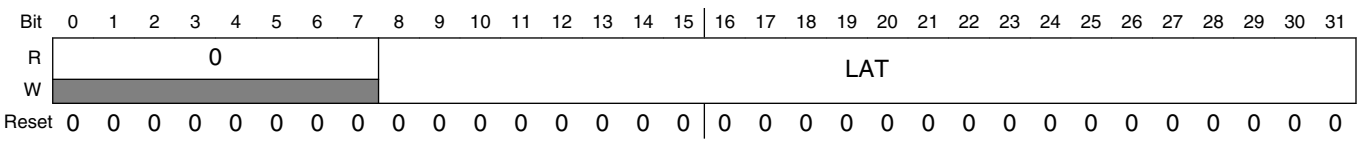
INTC_HIPRI*n*C2 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

28.5.23 INTC LAT Register (INTC_LAT*n*C2)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 107Ch offset + (4d × i), where i=0d to 2d



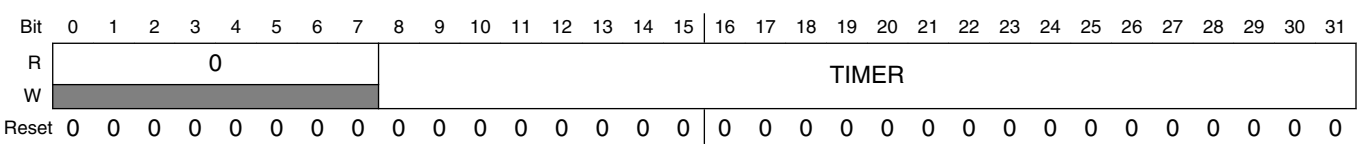
INTC_LAT*n*C2 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

28.5.24 INTC Timer Register (INTC_TIMER*n*C2)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 108Ch offset + (4d × i), where i=0d to 2d



INTC_TIMER*n*C2 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

28.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

28.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI*n* value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*n*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*n* is updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

28.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

28.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SELn] bits.

28.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–15 are assigned vectors 0–15, respectively. The peripheral interrupt requests are assigned vectors from 16 to a number as high as needed to cover all peripheral interrupt requests.

28.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*n*. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

28.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in Figure 28-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

28.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

28.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

28.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

28.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*n* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*n* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

28.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*n*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*n* does not need to be loaded from the associated INTC_CPR*n* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

Although the INTC supports up to 32 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 31 will not be preempted. Therefore, the LIFO supports the stacking of 31 priorities. However, the LIFO is only 30 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping

an empty LIFO are treated. If the LIFO is pushed 31 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

28.6.3 Handshaking with processor

This section describes handshaking with the processor.

28.6.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

28.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to Figure 28-3.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 28-3. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Software vector mode.

28.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*n*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*n* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

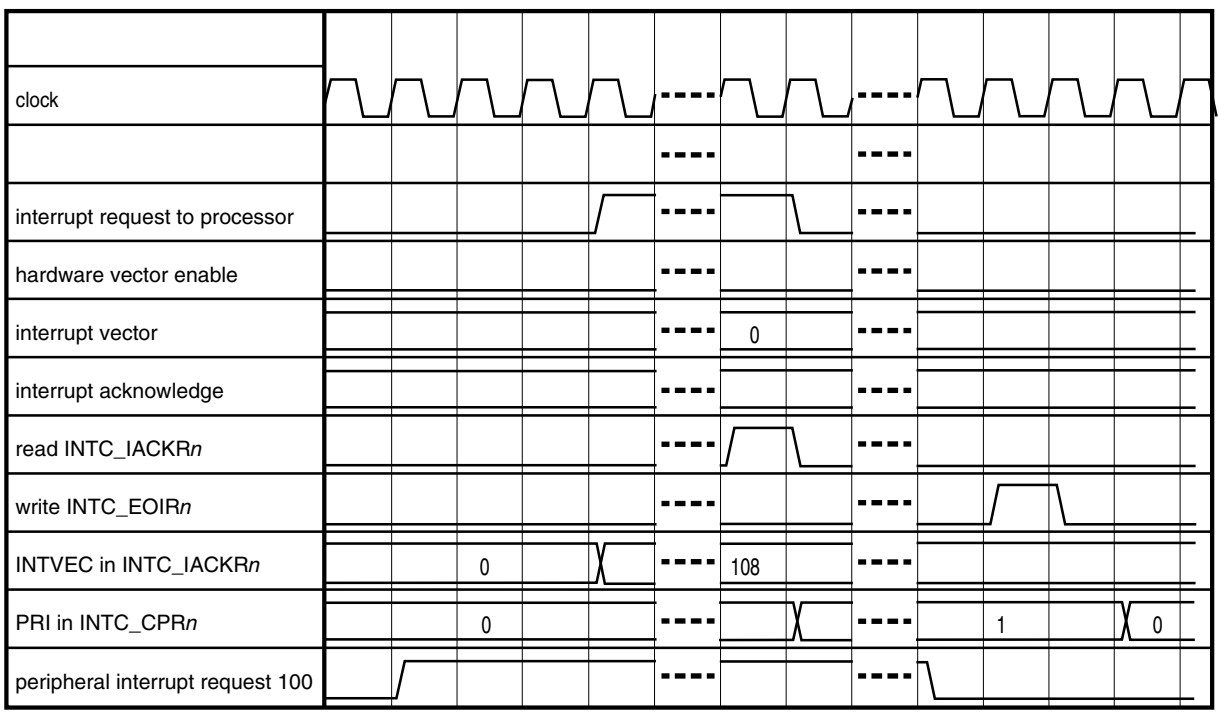


Figure 28-3. Timing diagram of software vector mode handshaking

28.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 28-4. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In

addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See End of interrupt exception handler.

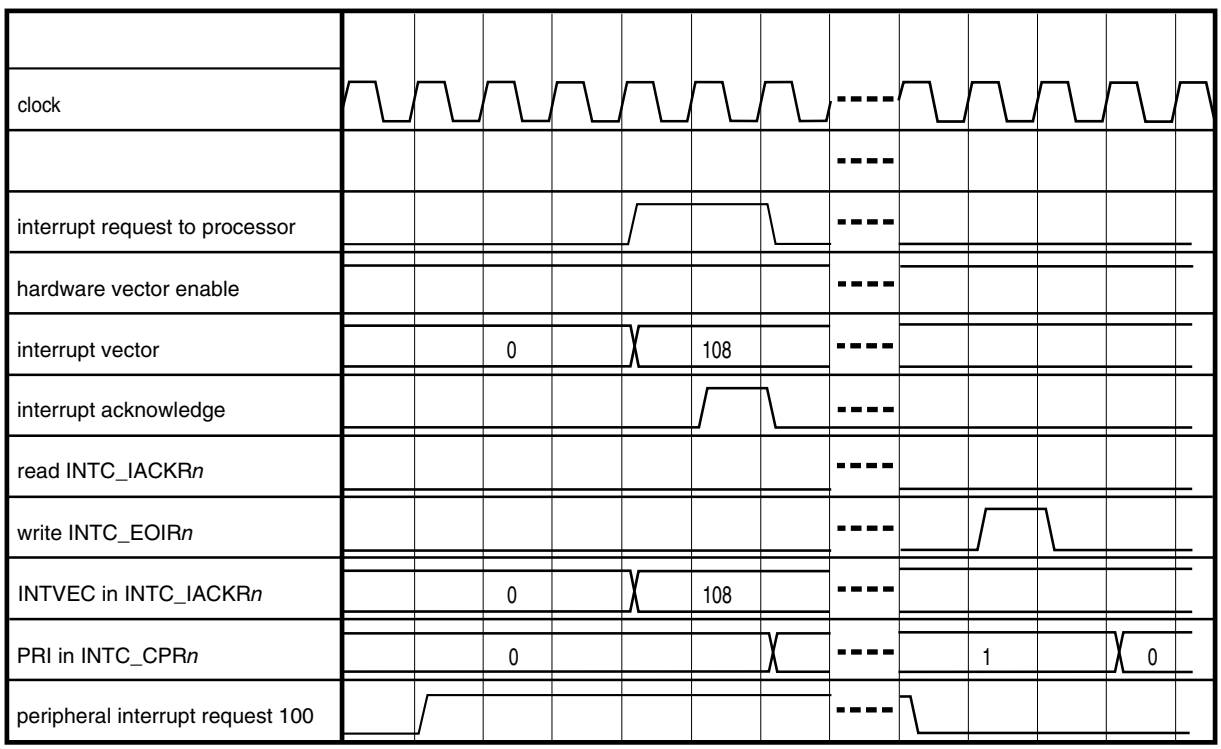


Figure 28-4. Timing diagram for hardware vector mode handshaking

28.6.4 Interrupt Controller Monitor (INTCM)

Isolated in its own hierarchy, the interrupt latency monitor logic is used to monitor or check interrupt latency for reliability, safety, and interrupt software requirements.

28.6.4.1 Operation

The following section lists the operation of the INTCM.

28.6.4.1.1 TIMERn

A clear of TIMERn will clear the corresponding error.

The TIMERn will not run if there is a corresponding error.

If the INTC_MMRn[MM] is enabled and an error is signaled, disable TIMERn.

If the INTC_MMRn[MM] is enabled, for a transition from 0 to 1 of SOURCE input that is being monitored by INTC_HIPRIn registers, clear and enable TIMERn. Once the timer reaches the maximum count, it will stop and not roll over. Software can read the TIMERn value at any time.

If the INTC_MMRn[MM] is enabled and if the TIMERn is enabled, then it should increment each cycle.

If the INTC_MMRn[MM] is enabled and if INTC_LATn is non-zero, an error will be signaled if the timer exceeds INTC_LATn or if the TIMERn reaches the maximum count.

If the INTC_MMRn[MM] is IACK and an interrupt acknowledge asserts for the monitored source, INTC_HIPRIn[IRQ], then the enabled TIMERn should stop. As an extra check, the IRQ is used to ensure that the acknowledged interrupt matches the monitored interrupt INTC_HIPRIn[IRQ].

If the INTC_MMRn[MM] is SW and an interrupt acknowledge occurs where a monitored interrupt is preempted, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is EOI or SW for a monitored interrupt which has an INTC_EOIRn, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is SW and if INTC_EOIRn, and if the previous one is monitored, it will be re-enabled.

28.7 Initialization/application information

This section contains initialization/application information.

28.7.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPRn registers is 31. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:

- Configure HVENn in INTC_BCR.
- Configure VTBAn in INTC_IACKRn.
- Raise the PRIn fields and set the PRC_SELn fields to the desired processor in INTC_PSRn.
- Set the enable bits or clear the mask bits for the peripheral interrupt requests.
- Lower PRI in INTC_CPRn to zero.
- Enable processor(s) recognition of interrupts.

28.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

28.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis        r3,INTC_IACKRn@ha        # form adjusted upper half of INTC_IACKRn address
lwz        r3,INTC_IACKRn@l(r3)     # load INTC_IACKRn, which clears request to processor
lwz        r3,0x0(r3)                # load address of ISR from vector table
wrteei    1                        # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlr    r3                        # move the INTC_IACKRn address into the link register
blrl                             # branch to ISR; link register updated with epilog
                                 # address
epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                             # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                  # form 0 to write to INTC_EOIRn
wrteei    0                       # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)        # store to INTC_EOIRn, informing INTC to lower priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
```

```
        .
        .
        .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)
blr                              # return to epilog
```

28.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b        interrupt_exception_handler_continuedn# 16 bytes available, branch to continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei    1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl        ISRn                # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                             # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                # form 0 to write to INTC_EOIRn
wrteei    0                    # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)    # store to INTC_EOIRn, informing INTC to lower priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

28.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs

execute above INTC_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*n* while the shared resource is being accessed.

An ISR whose PRI*n* in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

28.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 28-1. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4

Table continues on the next page...

Table 28-1. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

28.7.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

28.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*n*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority

ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

28.7.5.2 Ensuring coherency

This section discusses how to ensure coherency.

28.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*n*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*n* can be made by those system services with the code sequence:

```
GetResource:
        wrteei 0                # disable external interrupts to the Processor
        raise PRI                # Write to CPR, cache inhibit, guarded
        mbar                    # flush out writes from store buffer
        wrteei 1                # enable external interrupts to the Processor
        isync                    # re-fetch Processor pipeline

ReleaseResource:
        mbar                    # flush out writes from store buffer
        wrteei 0                # disable external interrupts to the Processor
        lower PRI                # Write to CPR, cache inhibit, guarded
        wrteei 1                # enable external interrupts to the Processor
```

28.7.5.2.1.1 Interrupt request to processor

Referencing Figure 28-1, the interrupt request logic to the processor is shown in the following figure.

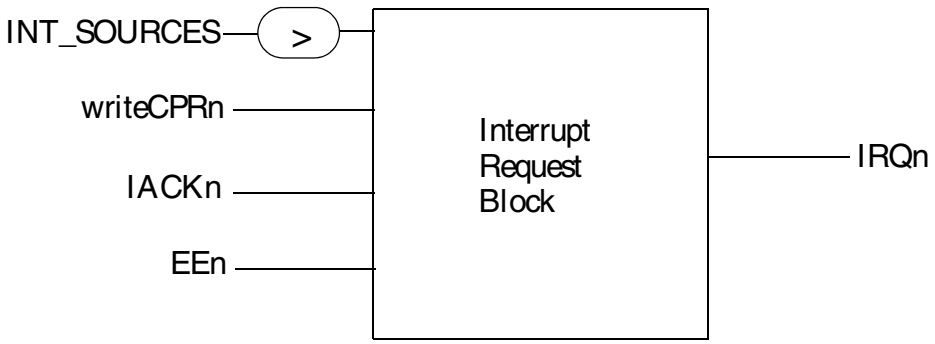


Figure 28-5. Interrupt Request Block Diagram

Assuming that there are no IRQn deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPRn, then interrupt request to the processor (IRQn) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EEn bit has been cleared by the wrteei instruction (see the code sequence in Interrupt with blocked priority), IRQn will be re-evaluated when processor n writes to the INTC_CPRn while processor n ignores IRQn

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPRn.

28.7.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. The shared coherent data block now can be accessed coherently. The following figure shows the timing diagram for this scenario, and the following table explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

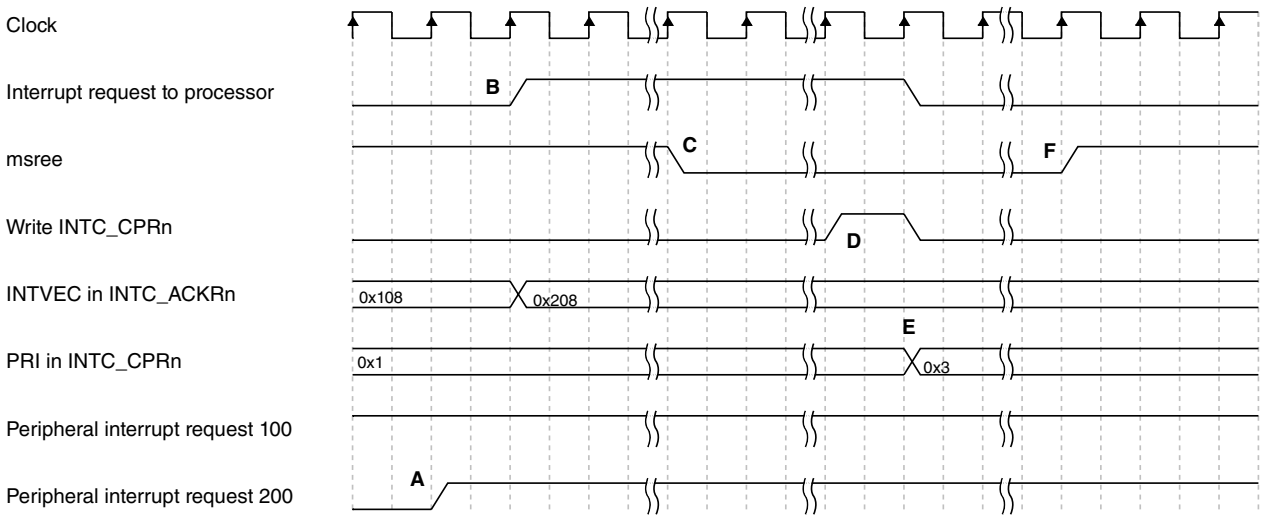


Figure 28-6. Timing diagram of raised priority preserved

Table 28-2. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

28.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 32 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

28.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

28.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRIn value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit

in INTC_SSCIR*n*. Writing a '1' to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

28.7.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

28.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

Lowering the PRI value in INTC_CPR*n* within an ISR to below the ISR's corresponding PRI value in INTC_PSR*n* allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

28.7.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

28.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

28.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

28.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to End of interrupt exception handler for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

28.7.10 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
      wrteei 0                # disable processor recognition of external interrupts
      oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)

pop_lifo:
      store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI, etc...
      examine INTC_CPRn[PRI], and store onto stack
      if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
      branch to push_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
      load stacked PRI value and store to INTC_CPRn
      load INTC_IACKn         # IACK; push INTC_CPRn into LIFO
      if stacked PRI values are not depleted, branch to push_lifo

      INTC_CPRn    = oldCPRn; # restore original INTC_CPRn and re-evaluate pending external
interrupts
      wrteei 1                # enable processor recognition of interrupts
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

28.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

Chapter 29
Direct Memory Access Multiplexer (DMAMUX)

29.1 DMA triggers to eDMA channels

The following diagram shows how the DMAMUXs trigger the eDMA.

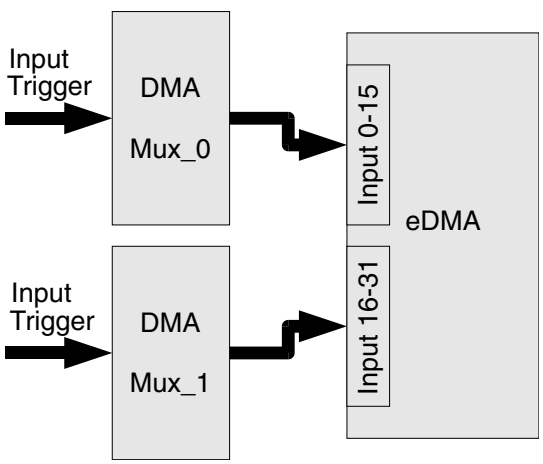


Figure 29-1. DMA triggers to eDMA channels

Table 29-1. DMAMUX to eDMA Channel Mapping

eDMA channel	DMAMUX output
DMAMUX_0 output 0 - 15	eDMA channel 0 - 15
DMAMUX_1 output 0 - 15	eDMA channel 16 - 31

29.2 Introduction

29.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

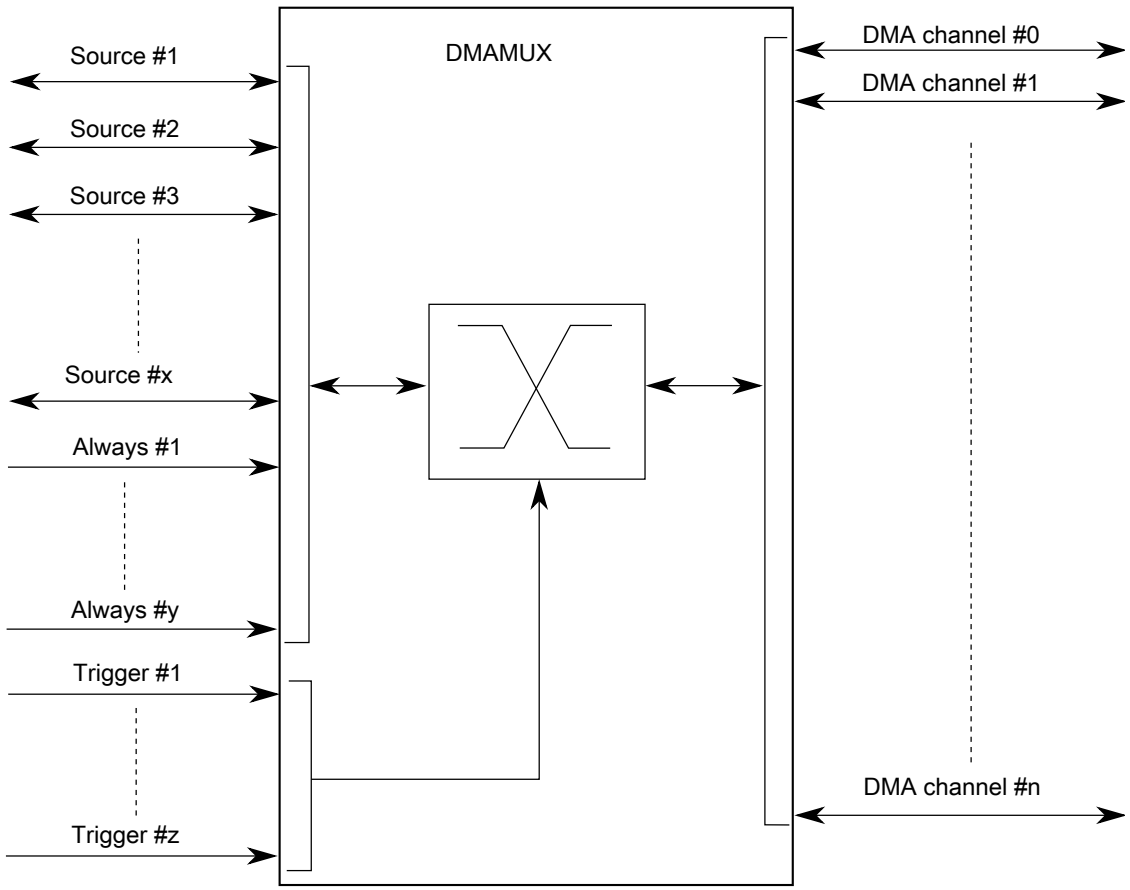


Figure 29-2. DMAMUX block diagram

29.2.2 Features

The DMAMUX module provides these features:

- Up to 34 peripheral slots and up to six always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first eight channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

29.2.3 Modes of operation

The following operating modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–7.

29.3 External signal description

The DMAMUX has no external pins.

29.4 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	29.4.1/1086
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	29.4.1/1086
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	29.4.1/1086
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	29.4.1/1086
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	29.4.1/1086

Table continues on the next page...

DMAMUX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	29.4.1/1086
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	29.4.1/1086
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	29.4.1/1086
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	29.4.1/1086
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	29.4.1/1086
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	29.4.1/1086
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	29.4.1/1086
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	29.4.1/1086
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	29.4.1/1086
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	29.4.1/1086
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	29.4.1/1086

29.4.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled

Table continues on the next page...

DMAMUX_CHCFG*n* field descriptions (continued)

Field	Description
1 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel. 0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

29.5 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

29.5.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 8 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

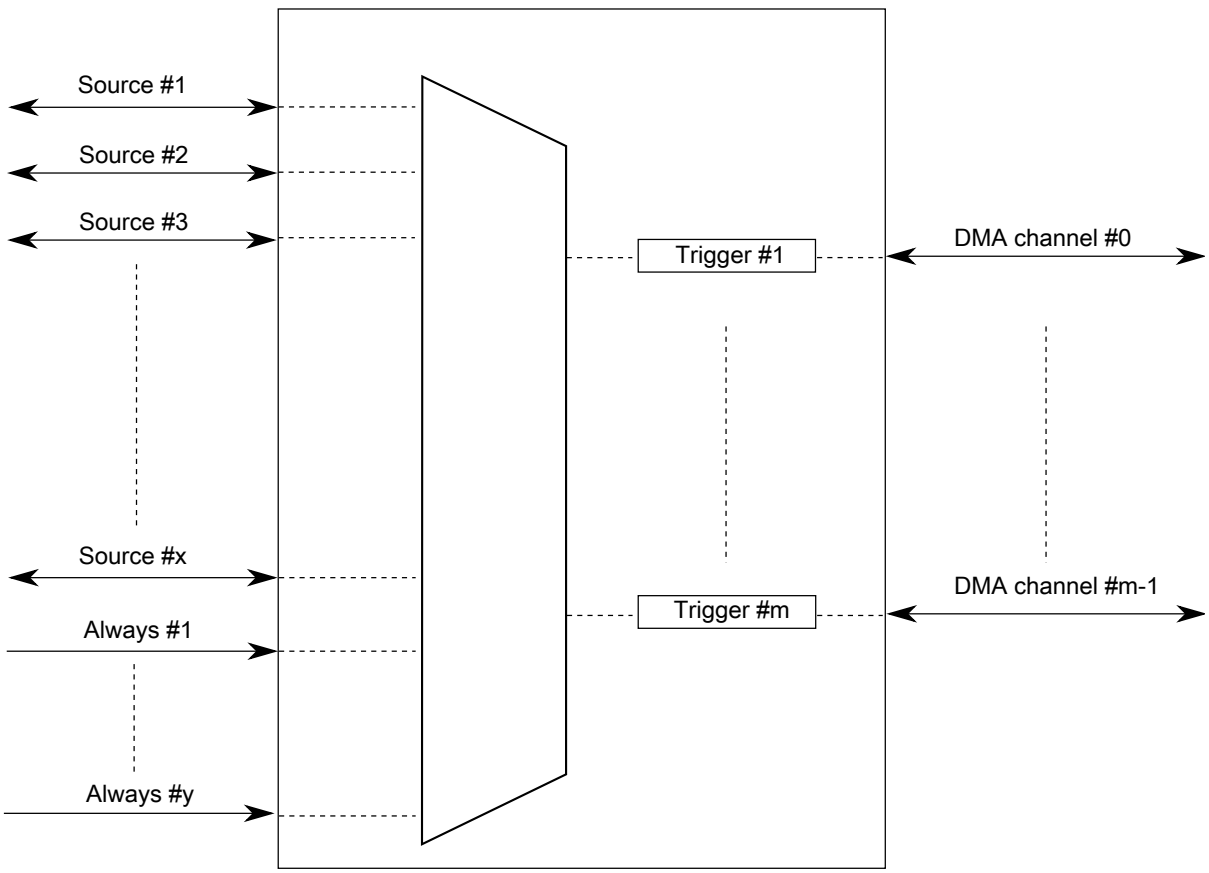


Figure 29-3. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

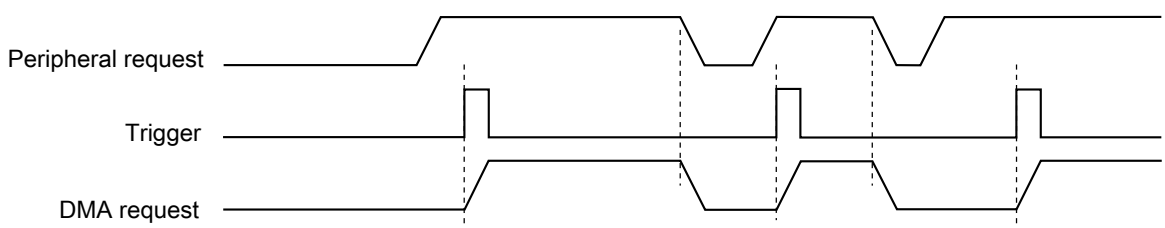


Figure 29-4. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

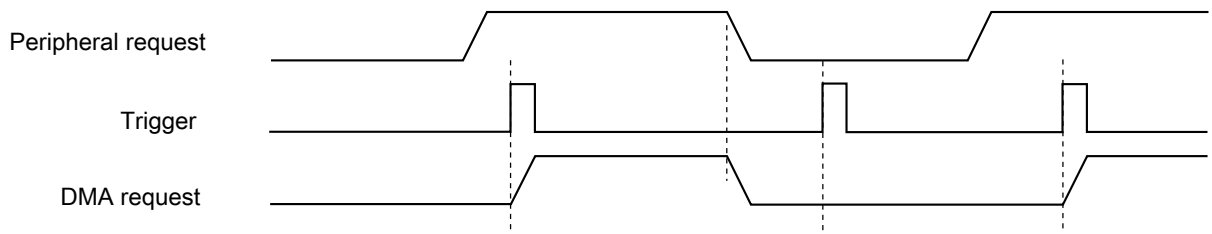


Figure 29-5. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

29.5.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

29.5.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are six additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

29.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

29.6.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

29.6.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 8 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1.

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 8 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
```

```
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 30
Platform Configuration Module (PCM)

30.1 Overview

The Platform Configuration Module contains miscellaneous configuration registers for the device. These configuration registers are related to the operation of the intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

The register at offset 00h is present but serves no purpose. This register can be written and read, but its contents do not affect any functionality and no error is returned from accessing this register.

30.2 Memory map and register definition

The Platform Configuration Module contains miscellaneous configuration registers for the device. These configuration registers are related to the operation of the intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

The register at offset 00h is present but serves no purpose. This register can be written and read, but its contents do not affect any functionality and no error is returned from accessing this register.

NOTE

PCM registers can only be accessed in supervisor mode.

PCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	IAHB Burst Enable 1 Register Register (PCM_IAHB_BE1)	32	R/W	0707_0707h	30.3.3/1096
8	IAHB Burst Enable 2F Register Register (PCM_IAHB_BE2)	32	R/W	0007_0707h	30.3.4/1098
C	IAHB Burst Enable 3 Register Register (PCM_IAHB_BE3)	32	R/W	0000_0707h	30.3.5/1099
10	IAHB Burst Enable 4 Register Register (PCM_IAHB_BE4)	32	R/W	0007_0707h	30.3.6/1101
14	IAHB Burst Enable 5 Register Register (PCM_IAHB_BE5)	32	R/W	0007_0707h	30.3.7/1103

30.3.3 IAHB Burst Enable 1 Register Register (PCM_IAHB_BE1)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					Reserved			0					PRE_ DMA	BRE_ DMA	BWE_DMA
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_ D2I	BRE_D2I	BWE_D2I	0					PRE_ I2D	BRE_I2D	BWE_I2D
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 Reserved	This field is reserved. Read/Write reserved bits that do not have any functionality
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_DMA	Pending read enable (PRE) DMA This bit controls the pending read transactions for the bus gasket for a concentrator that includes the DMA and SIPI. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_DMA	Burst read enable (BRE) DMA This bit controls the burst read transactions for the bus gasket for a concentrator that includes the DMA and SIPI.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_DMA	Burst write enable (BWE) DMA This bit controls the burst write transactions for the bus gasket for a concentrator that includes the DMA and SIPI. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_D2I	Pending read enable (PRE) D2I This bit controls the pending read transactions for the bus gasket for transactions going from the Data crossbar to the Instruction Crossbar. 0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_D2I	Burst read enable (BRE) D2I This bit controls the burst read transactions for the bus gasket for transactions going from the Data crossbar to the Instruction Crossbar. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_D2I	Burst write enable (BWE) D2I This bit controls the burst write transactions for the bus gasket for transactions going from the Data crossbar to the Instruction Crossbar. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_I2D	Pending read enable (PRE) I2D This bit controls the pending read transactions for the bus gasket for transactions going from the Instruction crossbar to the Data Crossbar. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_I2D	Burst read enable (BRE) I2D This bit controls the burst read transactions for the bus gasket for transactions going from the Instruction crossbar to the Data Crossbar. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_I2D	Burst write enable (BWE) I2D This bit controls the burst write transactions for the bus gasket for transactions going from the Instruction crossbar to the Data Crossbar.

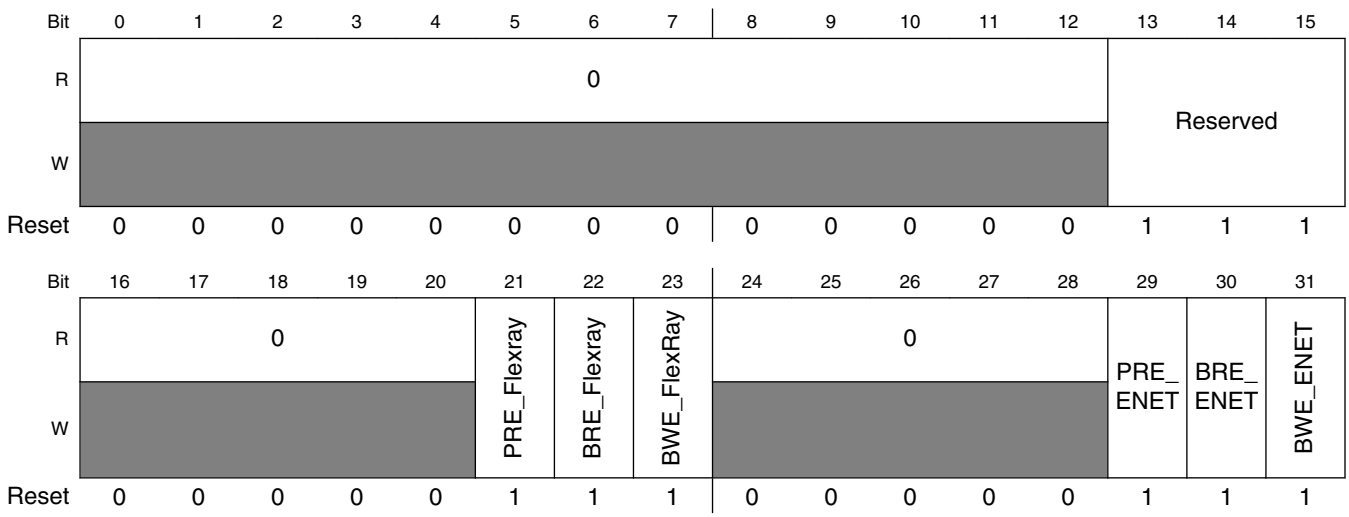
Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
	0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

30.3.4 IAHB Burst Enable 2F Register Register (PCM_IAHB_BE2)

Address: 0h base + 8h offset = 8h



PCM_IAHB_BE2 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 Reserved	This field is reserved. Read/Write reserved bits that do not have any functionality
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_Flexray	Pending read enable (PRE) Flexray This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_Flexray	Burst read enable (BRE) Flexray This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_FlexRay	Burst write enable (BWE) Flexray

Table continues on the next page...

PCM_IAHB_BE2 field descriptions (continued)

Field	Description
	This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_ENET	Pending read enable (PRE) FEC and Flexray This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending writes are enabled.
30 BRE_ENET	Burst read enable (BRE) ENET and Flexray This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_ENET	Burst write enable (BWE) ENET and Flexray This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

30.3.5 IAHB Burst Enable 3 Register Register (PCM_IAHB_BE3)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

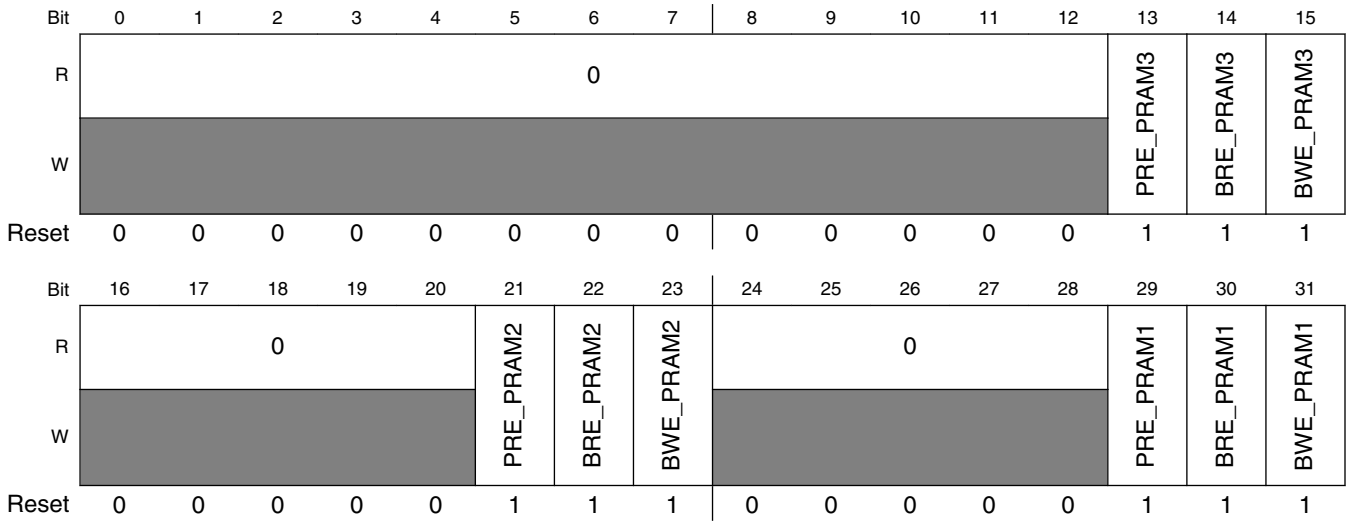
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_AIPS1	BRE_AIPS1	BWE_AIPS1	0					PRE_AIPS0	BRE_AIPS0	BWE_AIPS0
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE3 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_AIPS1	Pending read enable (PRE) AIPS1 This bit controls the pending read transactions for the bus gasket between the Data crossbar and the AIPS1. 0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_AIPS1	Burst read enable (BRE) AIPS1 This bit controls the burst read transactions for the bus gasket between the Data crossbar and the AIPS1. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_AIPS1	Burst write enable (BWE) AIPS1 This bit controls the burst write transactions for the bus gasket between the Data crossbar and the AIPS1. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_AIPS0	Pending read enable (PRE) AIPS0 This bit controls the pending read transactions for the bus gasket between the Data crossbar and the AIPS0. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_AIPS0	Burst read enable (BRE) AIPS0 This bit controls the burst read transactions for the bus gasket between the Data crossbar and the AIPS0. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_AIPS0	Burst write enable (BWE) AIPS0 This bit controls the burst write transactions for the bus gasket between the Data crossbar and the AIPS0. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

30.3.6 IAHB Burst Enable 4 Register Register (PCM_IAHB_BE4)

Address: 0h base + 10h offset = 10h



PCM_IAHB_BE4 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_PRAM3	Pending read enable (PRE) PRAM2 This bit controls the pending read transactions for the bus gasket between the Data crossbar and the pram_port2. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_PRAM3	Burst read enable (BRE) PRAM2 This bit controls the burst read transactions for the bus gasket between the Data crossbar and the pram_port2. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_PRAM3	Burst write enable (BWE) PRAM2 This bit controls the burst write transactions for the bus gasket between the Data crossbar and the pram_port2. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_PRAM2	Pending read enable (PRE) PRAM1 This bit controls the pending read transactions for the bus gasket between the Data crossbar and the pram_port1.

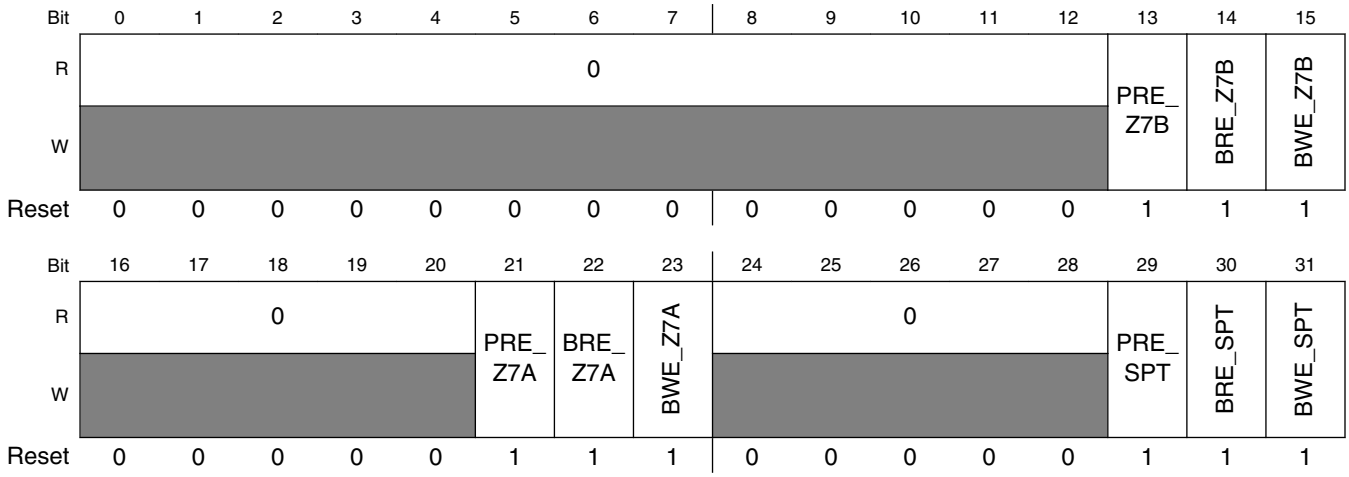
Table continues on the next page...

PCM_IAHB_BE4 field descriptions (continued)

Field	Description
	0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_PRAM2	Burst read enable (BRE) PRAM1 This bit controls the burst read transactions for the bus gasket between the Data crossbar and the pram_port1. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_PRAM2	Burst write enable (BWE) PRAM1 This bit controls the burst write transactions for the bus gasket between the Data crossbar and the pram_port1. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_PRAM1	Pending read enable (PRE) PRAM0 This bit controls the pending read transactions for the bus gasket between the Data crossbar and the pram_port0. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_PRAM1	Burst read enable (BRE) PRAM0 This bit controls the burst read transactions for the bus gasket between the Data crossbar and the pram_port0. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_PRAM1	Burst write enable (BWE) PRAM0 This bit controls the burst write transactions for the bus gasket between the Data crossbar and the pram_port0. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

30.3.7 IAHB Burst Enable 5 Register Register (PCM_IAHB_BE5)

Address: 0h base + 14h offset = 14h



PCM_IAHB_BE5 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_Z7B	Pending read enable (PRE) Z7B This bit controls the pending read transactions for the bus gasket between the Data crossbar and the z7b TCM interface. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_Z7B	Burst read enable (BRE) Z7B This bit controls the burst read transactions for the bus gasket between the Data crossbar and the z7b TCM interface. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_Z7B	Burst write enable (BWE) Z7B This bit controls the burst write transactions for the bus gasket between the Data crossbar and the z7b TCM interface. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_Z7A	Pending read enable (PRE) Z7A This bit controls the pending read transactions for the bus gasket between the Data crossbar and the z7a TCM interface.

Table continues on the next page...

PCM_IAHB_BE5 field descriptions (continued)

Field	Description
	0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_Z7A	Burst read enable (BRE) Z7A This bit controls the burst read transactions for the bus gasket between the Data crossbar and the z7a TCM interface. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_Z7A	Burst write enable (BWE) Z7A This bit controls the burst write transactions for the bus gasket between the Data crossbar and the z7a TCM interface. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_SPT	Pending read enable (PRE) SPT This bit controls the pending read transactions for the bus gasket between the SPT and the Data crossbar. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_SPT	Burst read enable (BRE) SPT This bit controls the burst read transactions for the bus gasket between the SPT and the Data crossbar. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_SPT	Burst write enable (BWE) SPT This bit controls the burst write transactions for the bus gasket between the SPT and the Data crossbar. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

Chapter 31
Multipurpose Control Block (MCB)

31.1 Memory Map and Registers

NOTE

Any access at address offsets 0x04h and between 0x2Ch to0x3Ch will not generate transfer error.

MCB memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	AFE Filter Enable Register (MCB_AFE_FILTEN)	32	R/W	See section	31.1.1/1106
8	NPC Special Enable Control (MCB_NPC_SPECIAL_ENABLE)	32	R/W	See section	31.1.2/1107
C	AFE LVD Mask (MCB_AFE_LVD_MASK)	32	R/W	0000_0000h	31.1.3/1108
10	Miscellaneous 1 register (MCB_MISC1)	32	R/W	0000_0000h	31.1.4/1110
14	Miscellaneous 2 Register (MCB_MISC2)	32	R/W	See section	31.1.5/1112
18	Miscellaneous 3 Register (MCB_MISC3)	32	R/W	See section	31.1.6/1114
1C	AFE Filter0 Phase Select Register (MCB_AFE_FILT0_PHASE)	32	R/W	See section	31.1.7/1114
20	AFE Filter1 Phase Select Register (MCB_AFE_FILT1_PHASE)	32	R/W	0000_0000h	31.1.8/1115
24	AFE Filter2 Phase Select (MCB_AFE_FILT2_PHASE)	32	R/W	See section	31.1.9/1116
28	AFE Filter3 Phase Select Register (MCB_AFE_FILT3_PHASE)	32	R/W	See section	31.1.10/ 1117
40	Nexus Trace Fifo Status Register (MCB_NEX_FIFO_STATUS)	32	R/W	See section	31.1.11/ 1118
4028	CLKOUT Source Select (MCB_CLKOUT_SEL)	32	R/W	0000_0002h	31.1.12/ 1119

31.1.1 AFE Filter Enable Register (MCB_AFE_FILTEN)

This register enables the AFE filter triggers. This register gets reset on every reset.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RESERVED															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED															GLOBAL_ FILTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

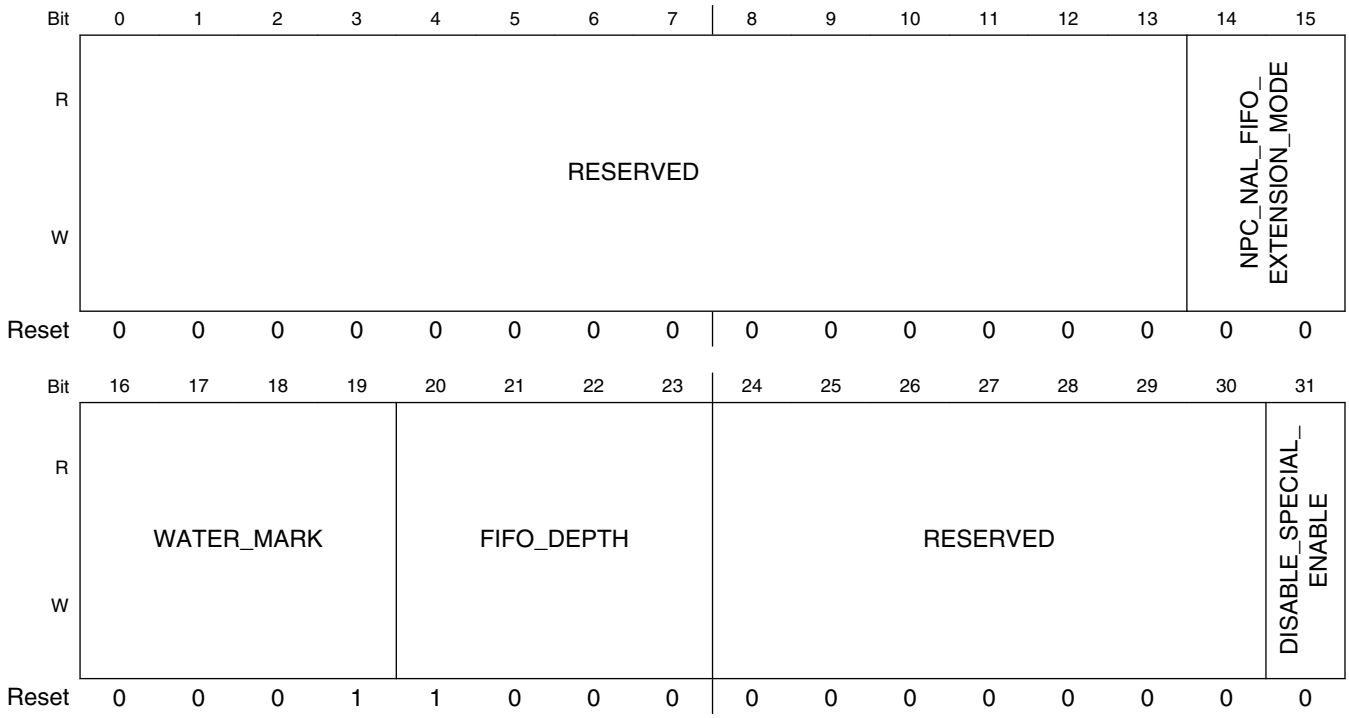
MCB_AFE_FILTEN field descriptions

Field	Description
0–30 RESERVED	This field is reserved.
31 GLOBAL_ FILTEN	Filter Enable 1 Filters are enabled once RFS is received 0 RFS is gated

31.1.2 NPC Special Enable Control (MCB_NPC_SPECIAL_ENABLE)

Enable/Disable special enable functionality. This register gets reset on every reset.

Address: 0h base + 8h offset = 8h



MCB_NPC_SPECIAL_ENABLE field descriptions

Field	Description
0–13 RESERVED	This field is reserved.
14–15 NPC_NAL_FIFO_EXTENSION_MODE	NPC NAL FIFO EXTENSION_MODE These bits are used to extend the fifo present between NPC and NAL from default 8 to 16. 00 Disable NAL_NPC Extended fifo mode 11 Enable NAL_NPC Extended fifo mode. Extends NAL existing fifo to make it 15 words deep. Set this bit prior to starting a NAL-NPC transaction. In this mode MCB_NPC_SPECIAL_ENABLE[WATER_MARK] should be configured to 4'b0101 or lower and MCB_NPC_SPECIAL_ENABLE[FIFO_DEPTH] should be programmed to 4'b1110
16–19 WATER_MARK	WATER MARK LEVEL NAL FIFO watermark level. This can be used to throttle trace from various nexus sources inside chip.
20–23 FIFO_DEPTH	NAL FIFO DEPTH Keep it always at 0x8 in default mode
24–30 RESERVED	RESERVED This field is reserved.

Table continues on the next page...

MCB_NPC_SPECIAL_ENABLE field descriptions (continued)

Field	Description
31 DISABLE_ SPECIAL_ ENABLE	Disable special enable fuinctionality of NPC auxilary ports Disables special enable functionality on NPC auxillary port related pads (such as MDOs, nex_ready_b, MCKO, EVTO, EVTI, MSEO) 0 Enable Special Enable functionality of NPC auxilary ports 1 Disable Special Enable fuinctionality of NPC auxilary ports

31.1.3 AFE LVD Mask (MCB_AFE_LVD_MASK)

This register gets reset on every reset.

Masks the eight LVDs from AFE. If not masked they act as source of destructive reset. By default the LVDs are masked.

NOTE

Before un-masking AFE LVD through the MCB register, the respective LVD status must be first polled from the relevant LVD status source. Only if LVD is in the released state should it be un-masked.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RESERVED															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RESERVED						AFE_LVD9_MASK	AFE_LVD8_MASK	AFE_LVD7_MASK	AFE_LVD6_MASK	AFE_LVD5_MASK	AFE_LVD4_MASK	AFE_LVD3_MASK	AFE_LVD2_MASK	Reserved	AFE_REF_LVD_MASK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_AFE_LVD_MASK field descriptions

Field	Description
0–21 RESERVED	This field is reserved.

Table continues on the next page...

MCB_AFE_LVD_MASK field descriptions (continued)

Field	Description
22 AFE_LVD9_ MASK	Mask LVD on AFE DAC internal capacitor circuitry The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE DAC internal capacitor circuitry 0 Mask LVD on AFE DAC internal capacitor circuitry
23 AFE_LVD8_ MASK	Mask LVD on level shifter logic The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on level shifter logic 0 Mask LVD on level shifter logic
24 AFE_LVD7_ MASK	Mask LVD on AFE SDPLL digital circuitry The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE SDPLL digital circuitry 0 Mask LVD on AFE SDPLL digital circuitry
25 AFE_LVD6_ MASK	Mask LVD on AFE SDPLL analog circuitry The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE SDPLL analog circuitry 0 Mask LVD on AFE SDPLL analog circuitry
26 AFE_LVD5_ MASK	Mask LVD on AFE OSC circuitry The user must write '1' to make LVDs the source of destructive reset 1 Not Mask LVD on AFE OSC circuitry 0 Mask LVD on AFE OSC circuitry
27 AFE_LVD4_ MASK	Mask LVD on AFE DAC external capacitor circuitry The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE DAC external capacitor circuitry 0 Mask LVD on AFE DAC external capacitor circuitry
28 AFE_LVD3_ MASK	Mask LVD on AFE ADC digital circuitry The user must write '1' to make LVDs the source of destructive reset 1 Not Mask LVD on AFE ADC digital circuitry 0 Mask LVD on AFE ADC digital circuitry
29 AFE_LVD2_ MASK	Mask LVD on AFE ADC analog circuitry The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE ADC analog circuitry 0 Mask LVD on AFE ADC analog circuitry
30 Reserved	This field is reserved.

Table continues on the next page...

MCB_AFE_LVD_MASK field descriptions (continued)

Field	Description
31 AFE_REF_LVD_MASK	Mask LVD on VDD_HV_RAW and VDD_HV_DAC The user must write '1' to make LVDs the source of destructive reset. 1 Not Mask LVD on AFE reference voltage 0 Mask LVD on AFE reference voltage

31.1.4 Miscellaneous 1 register (MCB_MISC1)

This register gets reset on every reset.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															SIN_END_BYP
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved									TS_CLK_SEL	OUT_CTE	IN_CTE	Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_MISC1 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 SIN_END_BYP	single ended bypass 0 XOSC Single Ended Bypass mode is not used. 1 XOSC Single Ended Bypass mode is used.
16–24 Reserved	This field is reserved.
25 TS_CLK_SEL	Select signal for ENET Time Clock 1 Clock source is PAD_94 0 Clock source is Aux clock 11 (AC11)
26 OUT_CTE	To select the RCS/RFS outputs from CTE.

Table continues on the next page...

MCB_MISC1 field descriptions (continued)

Field	Description
	0 From PAD. Refer the "IO Signal Description and Input multiplexing tables" spreadsheet attached with this document. 1 HSYNC/VSYNC from MIPICSI-2
27 IN_CTE	To select the RCS/RFS inputs to CTE. 0 From PAD. Refer the "IO Signal Description and Input multiplexing tables" spreadsheet attached with this document. 1 HSYNC/VSYNC from MIPICSI-2
28–31 Reserved	This field is reserved.

31.1.5 Miscellaneous 2 Register (MCB_MISC2)

This register gets reset on every reset. When Nexus Trace is not used, then user must configure MCB_MISC2 with value 0x0 prior to using SPT.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				NXMC_2_FIFO_EN	Reserved			SPT_NEXUS_THROTTLE_CONTROL						RCCU_DMA_STATUS	RCCU_Z4_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*

* Notes:

- RCCU_Z4_STATUS field: The reset value depends on dcf client DCF_SOC_CONF1 UTEST dcf client.

- RCCU_DMA_STATUS field: The reset value depends on dcf client DCF_SOC_CONF1 UTEST dcf client.

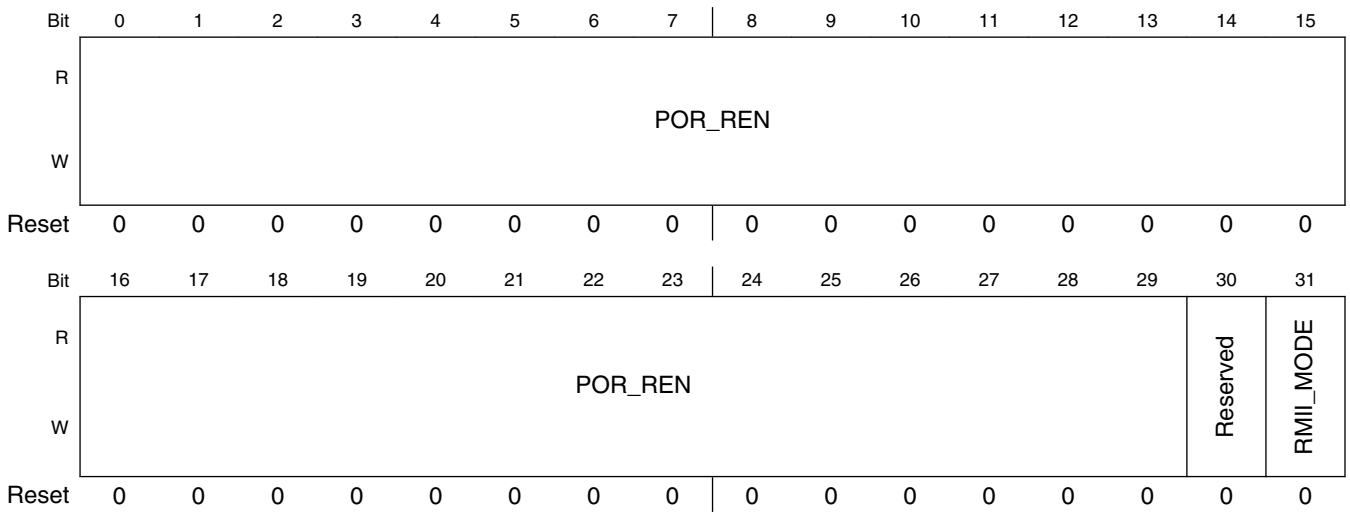
MCB_MISC2 field descriptions

Field	Description
0–19 Reserved	This field is reserved.
20 NXMC_2_FIFO_EN	Enable signal for NXMC_2 FIFO 0 NXMC_2 snoops SPT data directly 1 SPT AHB interface signals are captured in a FIFO. NXMC_2 receives data from this FIFO's output interface.
21–23 Reserved	Reserved This field is reserved.
24–29 SPT_NEXUS_THROTTLE_CONTROL	SPT NEXUS THROTTLE CONTROL These bits may be written to control the bandwidth requirement of SPT related NXMC client. For more information refer "Avoiding Trace Overflows" section in the "Chip-specific NXMC information" section.
30 RCCU_DMA_STATUS	RCCU DMA STATUS This bit shows DMA RCCU active status. Write 0 to this bit to clear the status, however if the DMA RCCU is active, this bit will get set again. 0 DMA RCCU is not active 1 DMA RCCU is active
31 RCCU_Z4_STATUS	RCCU Z4 STATUS This bit shows the Z4 RCCU active status. Write 0 to this bit to clear the status, however if the Z4 RCCU is active, this bit will get set again. 0 Z4 RCCU is not active 1 Z4 RCCU is active

31.1.6 Miscellaneous 3 Register (MCB_MISC3)

This register gets reset on POR only.

Address: 0h base + 18h offset = 18h



MCB_MISC3 field descriptions

Field	Description
0–29 POR_REN	Power retention These maintain their value across any reset other than POR.
30 Reserved	This field is reserved. This bit should be always written with its default value.
31 RMII_MODE	RMII MODE SELECTION Select between RMII clock to be output or input 0 RMII clock in input 1 RMII Clock is output

31.1.7 AFE Filter0 Phase Select Register (MCB_AFE_FILT0_PHASE)

AFE FILTER0 PHASE SELECT

This register gets reset on POR only.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved									MODE	PHASE					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_AFE_FILT0_PHASE field descriptions

Field	Description
0–24 Reserved	This field is reserved.
25 MODE	MODE See AFE FILTER ENABLE. Refer to Figure 48-1 0 Phase 0 starts one cycle (of CTE protocol clock) after RFS assertion. 1 Phase 0 starts along with RFS
26–31 PHASE	PHASE Choose between one of 64 phases. The delay is in terms of CTE protocol clock.

31.1.8 AFE Filter1 Phase Select Register (MCB_AFE_FILT1_PHASE)

AFE FILTER1 PHASE SELECT

This register gets reset on POR only.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved									MODE	PHASE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_AFE_FILT1_PHASE field descriptions

Field	Description
0–24 Reserved	This field is reserved.
25 MODE	MODE See AFE FILTER ENABLE. Refer to Figure 48-1 0 Phase 0 starts one cycle (of CTE protocol clock) after RFS assertion. 1 Phase 0 starts along with RFS
26–31 PHASE	PHASE Choose between one of 64 phases. The delay is in terms of CTE protocol clock.

31.1.9 AFE Filter2 Phase Select (MCB_AFE_FILT2_PHASE)

AFE FILTER2 PHASE SELECT

This register gets reset on POR only.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved									MODE	PHASE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_AFE_FILT2_PHASE field descriptions

Field	Description
0–24 Reserved	This field is reserved.
25 MODE	MODE See AFE FILTER ENABLE. Refer to Figure 48-1 0 Phase 0 starts one cycle (of CTE protocol clock) after RFS assertion. 1 Phase 0 starts along with RFS
26–31 PHASE	PHASE Choose between one of 64 phases. The delay is in terms of CTE protocol clock.

31.1.10 AFE Filter3 Phase Select Register (MCB_AFE_FILT3_PHASE)

AFE FILTER3 PHASE SELECT

This register gets reset on POR only.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved									MODE	PHASE					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_AFE_FILT3_PHASE field descriptions

Field	Description
0–24 Reserved	This field is reserved.
25 MODE	MODE See AFE FILTER ENABLE. Refer to Figure 48-1 0 Phase 0 starts one cycle (of CTE protocol clock) after RFS assertion. 1 Phase 0 starts along with RFS
26–31 PHASE	PHASE Choose between one of 64 phases. The delay is in terms of CTE protocol clock.

31.1.11 Nexus Trace Fifo Status Register (MCB_NEX_FIFO_STATUS)

This register gets reset on POR only.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved														NAL_NPC_FIFO_OFLOW	NXMC_2_FIFO_OFLOW
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCB_NEX_FIFO_STATUS field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 NAL_NPC_ FIFO_OFLOW	Overflow status of NAL_NPC Extended fifo 0 No overflow has occured in NAL_NPC Extended fifo. 1 NAL_NPC Extended fifo has overflown.
31 NXMC_2_FIFO_ OFLOW	Overflow status of NXMC_2_FIFO 0 No overflow has occured in NXMC_2_FIFO. 1 NXMC_2_FIFO has overflown.

31.1.12 CLKOUT Source Select (MCB_CLKOUT_SEL)

Select possible sources for clockout0 and clkout1

NOTE

CTE divided clock can only be observed when CTECK_DV field of CTE_CNTRL1 is programmed greater than 1.

This register gets reset on every reset.

Address: 0h base + 4028h offset = 4028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RESERVED								CGM_SINK_SEL_3				CGM_SINK_SEL2			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CGM_SINK_SEL_1				RESERVED	RESERVED	CLKOUT0		CLKOUT1						RESERVED	RESERVED
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

MCB_CLKOUT_SEL field descriptions

Field	Description
0–7 RESERVED	This field is reserved.
8–11 CGM_SINK_ SEL_3	Clocks mux selection For MCB_CLKOUT_SEL[CLKOUT 0] - Clock selected by CGM_SINK_SEL_3 For MCB_CLKOUT_SEL[CLKOUT 1] - Clock selected by CGM_SINK_SEL_3

Table continues on the next page...

MCB_CLKOUT_SEL field descriptions (continued)

Field	Description
	0000 SYS_CLK 0001 CORE1_CLK 0010 CORE2_CLK 0011 PBRIDGE_0/1_CLK 0100 DMA_CLK 0101 COREO_CLK 0110 SYS_CLK 0111 SYS_CLK 1000 PBRIDGE_0/1_CLK 1001 PBRIDGE_0/1_CLK 1010 PBRIDGE_0/1_CLK 1011 Core_0/Core_1 TCK 1100 Checker_Core_0/Core_2 TCK 1101 Flash clock 1110 Reserved 1111 Enable signal to flash to generate internal clock based on flash clock
12–15 CGM_SINK_SEL2	Clocks mux selection For MCB_CLKOUT_SEL[CLKOUT 0] - Clock selected by CGM_SINK_SEL_2 For MCB_CLKOUT_SEL[CLKOUT 1] - Clock selected by CGM_SINK_SEL_2 0000 ADC_CLK 0001 ADC_CLK 0010 Reserved 0011 CAN_CLK_OUT 0100 SPT_CLK 0101 LFAST_CLK 0110 ENET_CLK 0111 ENET_TIME_CLK 1000 SPI_CLK 1001 LIN_CLK 1010 Core_0 m_clk/nex_clk 1011 Core_0 slv_clk 1100 Core_1 m_clk/nex_clk 1101 Core_1 slv_clk 1110 Core_2 m_clk/nex_clk 1111 Core_2 slv_clk
16–19 CGM_SINK_SEL_1	Clocks mux selection For MCB_CLKOUT_SEL[CLKOUT 0] - Clock selected by CGM_SINK_SEL_1 For MCB_CLKOUT_SEL[CLKOUT 1] - Clock selected by CGM_SINK_SEL_1 0000 CGM CLK_AUX[0] 0001 CGM CLK_AUX[1] 0010 CGM CLK_AUX[2] 0011 CGM CLK_AUX[3] 0100 CGM CLK_AUX[4] 0101 Reserved

Table continues on the next page...

MCB_CLKOUT_SEL field descriptions (continued)

Field	Description
	0110 Reserved 0111 CGM CLK_AUX[7] 1000 CGM CLK_AUX[8] 1001 Reserved 1010 CGM CLK_AUX[10] 1011 CGM CLK_AUX[11] 1100 CGM CLK_AUX[12] 1101 CGM CLK_AUX[13] 1110 Reserved 1111 CGM CLK_SYS
20 RESERVED	This field is reserved.
21 RESERVED	This field is reserved. Always write the reset value to this field.
22–25 CLKOUT0	Clock Out 0000 RCOSC Clock 0001 XOSC Clock (Test Mode use of XOSC_CLK) 0010 PLL0 PHI Clock 0011 AFE 160 MHz Clock (NOTE: divide in MC_AUX_DIV before use) 0100 PLL1 PHI Clock 0101 NAP/NAL Clock 0110 OSC_40MHz (XOSC_CLK) 0111 Reserved 1000 CTE divided Clock 1001 PLL0 FBCLOCKIN IN PLL 1:1 mode 1010 AFE 320 MHZ Clock 1011 AFE 80 MHZ Clock 1100 Reserved 1101 Clock selected by CGM_SINK_SEL_1 1110 Clock selected by CGM_SINK_SEL_2 1111 Clock selected by CGM_SINK_SEL_3
26–29 CLKOUT1	Same options as for CLKOUT0
30 RESERVED	This field is reserved. Always write the reset value to this field.
31 RESERVED	This field is reserved.

Chapter 32
Dual PLL Digital Interface (PLLDIG)

32.1 Introduction

The Dual PLL system composed of PLL0 and PLL1 analog blocks and the digital interface (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1.

A key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1.

32.2 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

32.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
    - The loss of lock indication is sent to the FCCU via the system glue logic.

32.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The various modes of operation for the PLL are:

- Bypass mode with reference, and both PLLs disabled
- Normal mode with reference, and either PLL0 or both PLLs enabled

NOTE

Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<*mode*>_MC).

Bypass mode is defined as the mode where all clocks are driven by an external clock driving the EXTAL signal. Normal mode is defined as the mode where the clocks are driven by the PLLs. When using a crystal for the reference clock, reset should remain asserted until the oscillator has stabilized.

When PLL0:PHI1 is the source for PLL1, and PLL0 is powered down and working in bypass mode, PLL1 should be bypassed and powered down. PLL1 should be configured to work in normal mode only after PLL0 has achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

32.4.1 Bypass mode with reference, both PLLs disabled

In this mode, the PLLs are bypassed and system and peripheral clocks (PLL clock outputs) are driven from the external clock reference. The user must supply a clock that is within the appropriate frequency range. This mode is indicated by PLL0CR[CLKCFG] = 00b and PLL1CR[CLKCFG] = 00b. This is also the default power up mode of the PLLs.

32.4.2 Normal mode with reference, PLL0 or both PLLs enabled

In normal mode, PLL0 receives an input clock from the reference which can be prescaled by the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0 PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, and modulation enable are programmed by configuring the PLLFM register.

NOTE

While powering down the PLLs and if PLL0:PHI1 is used as the source for PLL1, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Configuring the appropriate MC_ME_<*mode*>_MC register to start PLL1 should only be done after PLL0 has achieved lock.

32.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

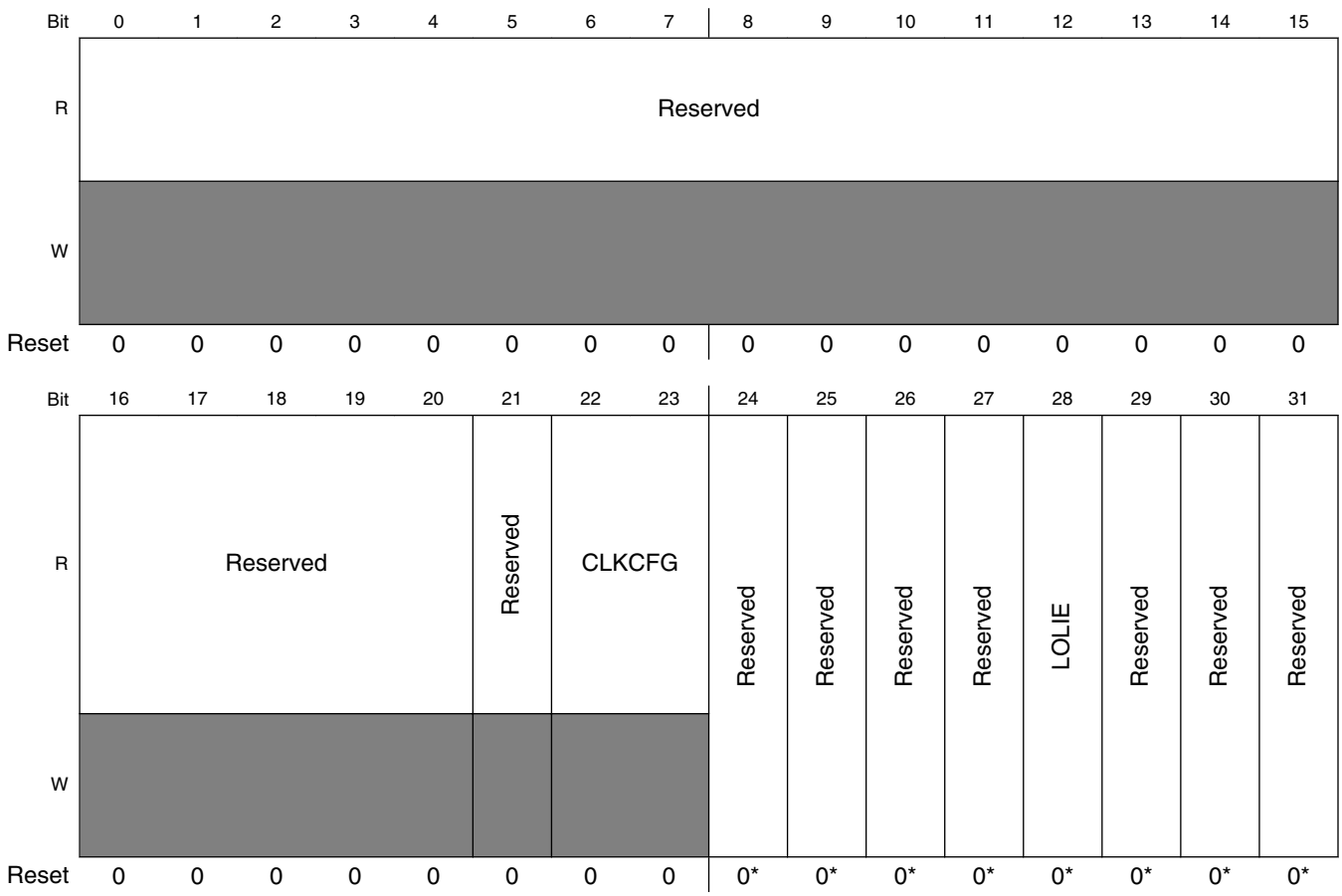
PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)	32	R/W	See section	32.5.1/1126
4	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	32	R/W	See section	32.5.2/1128
8	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)	32	R/W	See section	32.5.3/1130
20	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)	32	R/W	See section	32.5.4/1132
24	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	32	R/W	See section	32.5.5/1134
28	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)	32	R/W	See section	32.5.6/1135
2C	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)	32	R/W	0000_0000h	32.5.7/1137
30	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)	32	R/W	0000_0000h	32.5.8/1139

32.5.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)

The PLL0CR is shown in the following table.

Address: 0h base + 0h offset = 0h



* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: See the "Clocking" chapter for details on configuring this field.

PLLDIG_PLL0CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of PLL.

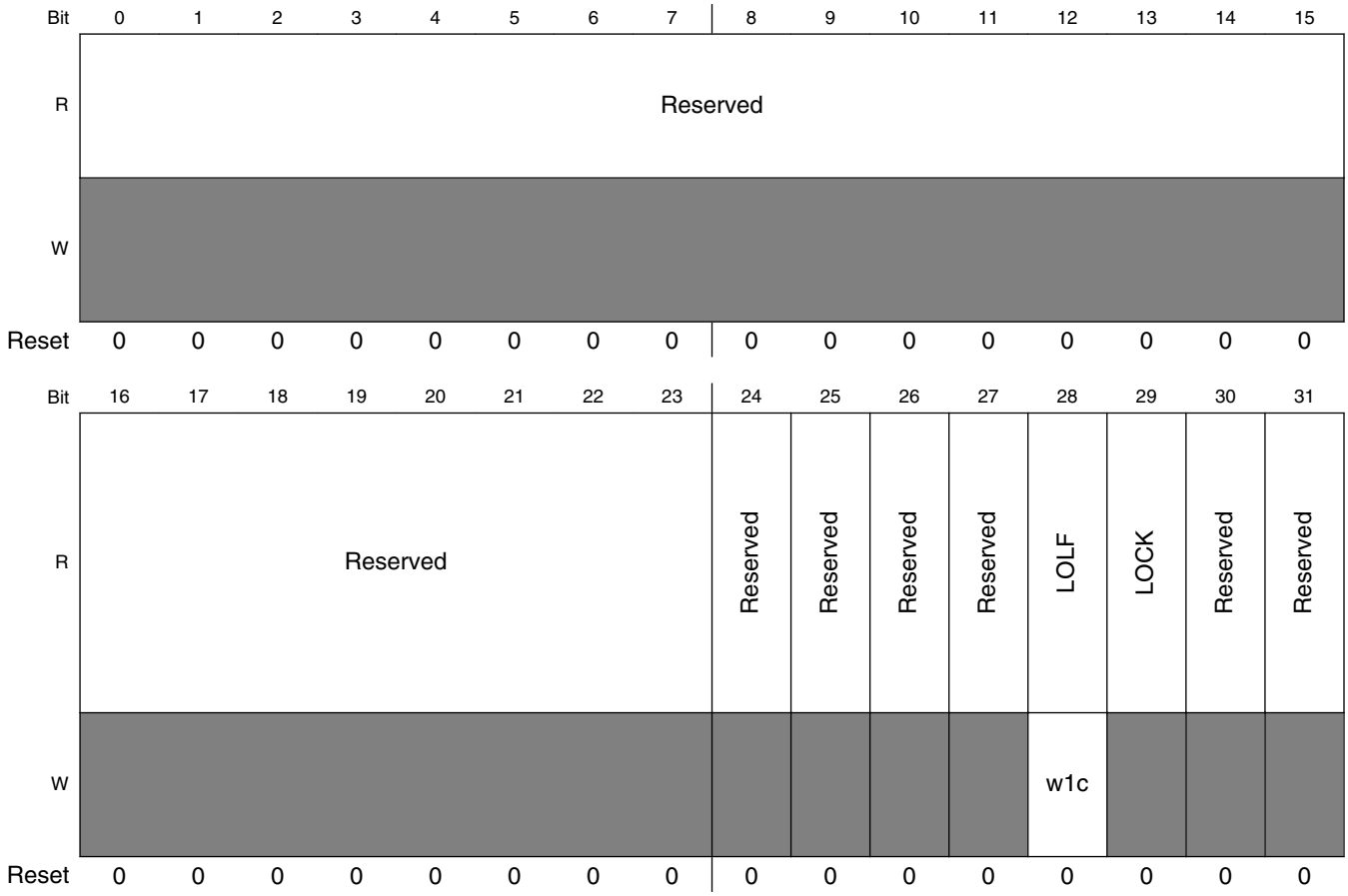
Table continues on the next page...

PLLDIG_PLL0CR field descriptions (continued)

Field	Description
	When PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1. **NOTE:** CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 Bypass mode with PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode, therefore LOLIE bit has no effect in bypass mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

32.5.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)

Address: 0h base + 4h offset = 4h



PLLDIG_PLL0SR field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. LOLF is not asserted if the loss-of-lock

Table continues on the next page...

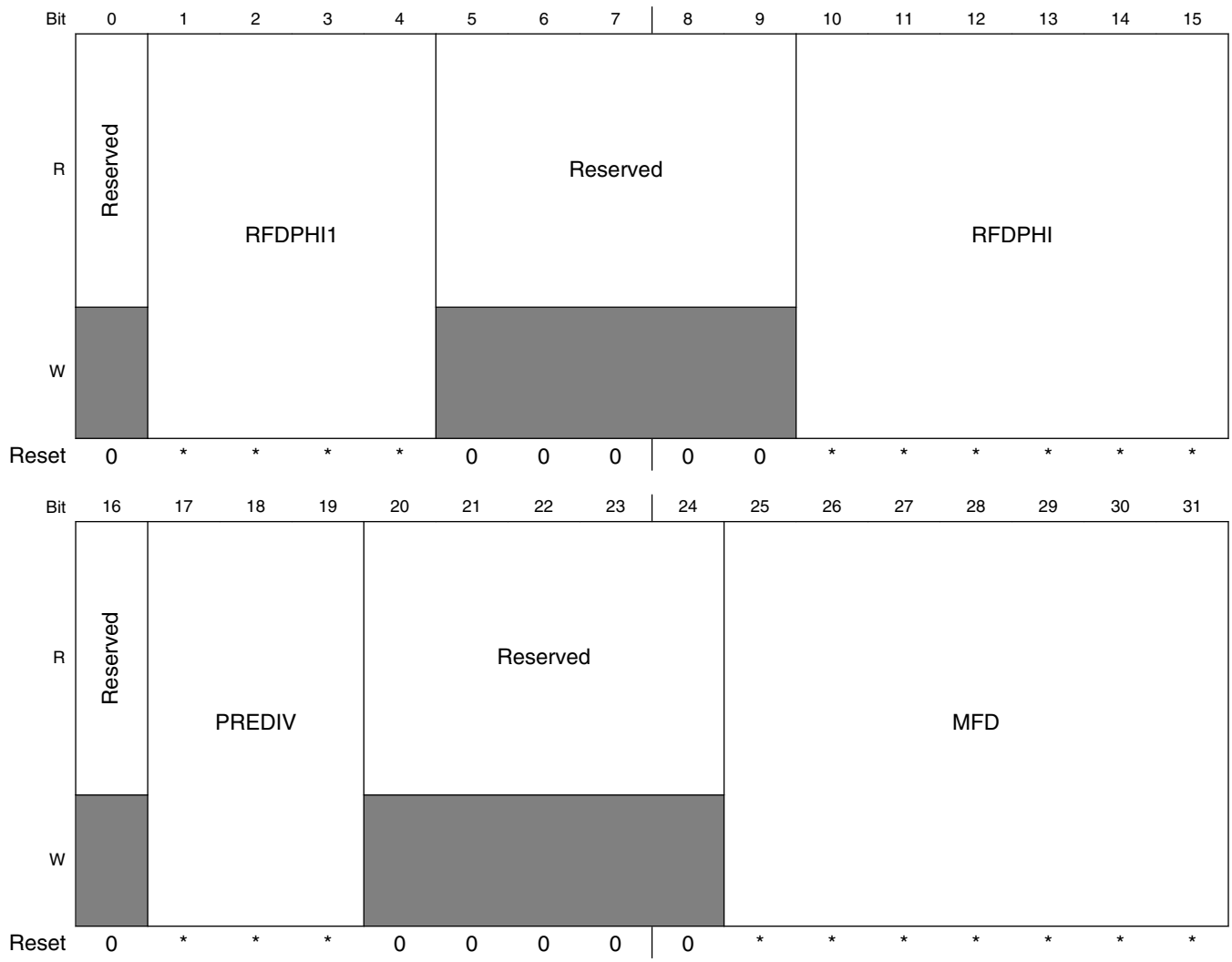
PLLDIG_PLL0SR field descriptions (continued)

Field	Description
	condition was detected while the PLL is in bypass mode, but changing from normal to bypass mode will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. If operating in bypass mode, the LOCK bit will still operate normally (for example, assert when the PLL acquires lock or negate when it loses lock). 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

32.5.3 PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency divider (RFDPHI and RFDPHI1), pre-divider (PREDIV), and loop divider (MFD). If the PLL0DV fields are modified without powering down the PLL, the PLL will lose lock and generate either a reset or interrupt based on which is enabled.

Address: 0h base + 8h offset = 8h



* Notes:
- MFD field: See Clocking chapter for reset value
- PREDIV field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value
- RFDPHI1 field: See Clocking chapter for reset value

PLLDIG_PLL0DV field descriptions

Field	Description
0 Reserved	This field is reserved.
1–4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO clock post divider for driving the PHI1 output clock. RFDPHI1 has a range of 4..15 (4h..Fh). All other values are reserved.
5–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16 Reserved	This field is reserved.
17–19 PREDIV	Input clock predivider. This field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Divide by 7
20–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 8...127 (8h...7Fh). All other values are reserved (see Clock configuration for details).

32.5.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)

The PLL1CR is shown in the following table. It contains the operational state of PLL1, and is used to enable/disable PLL specific interrupts.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					Reserved	CLKCFG		Reserved	Reserved	Reserved	Reserved	LOLIE	Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: Please refer to device "Clocking" chapter for details on configuring this field.

PLLDIG_PLL1CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration

Table continues on the next page...

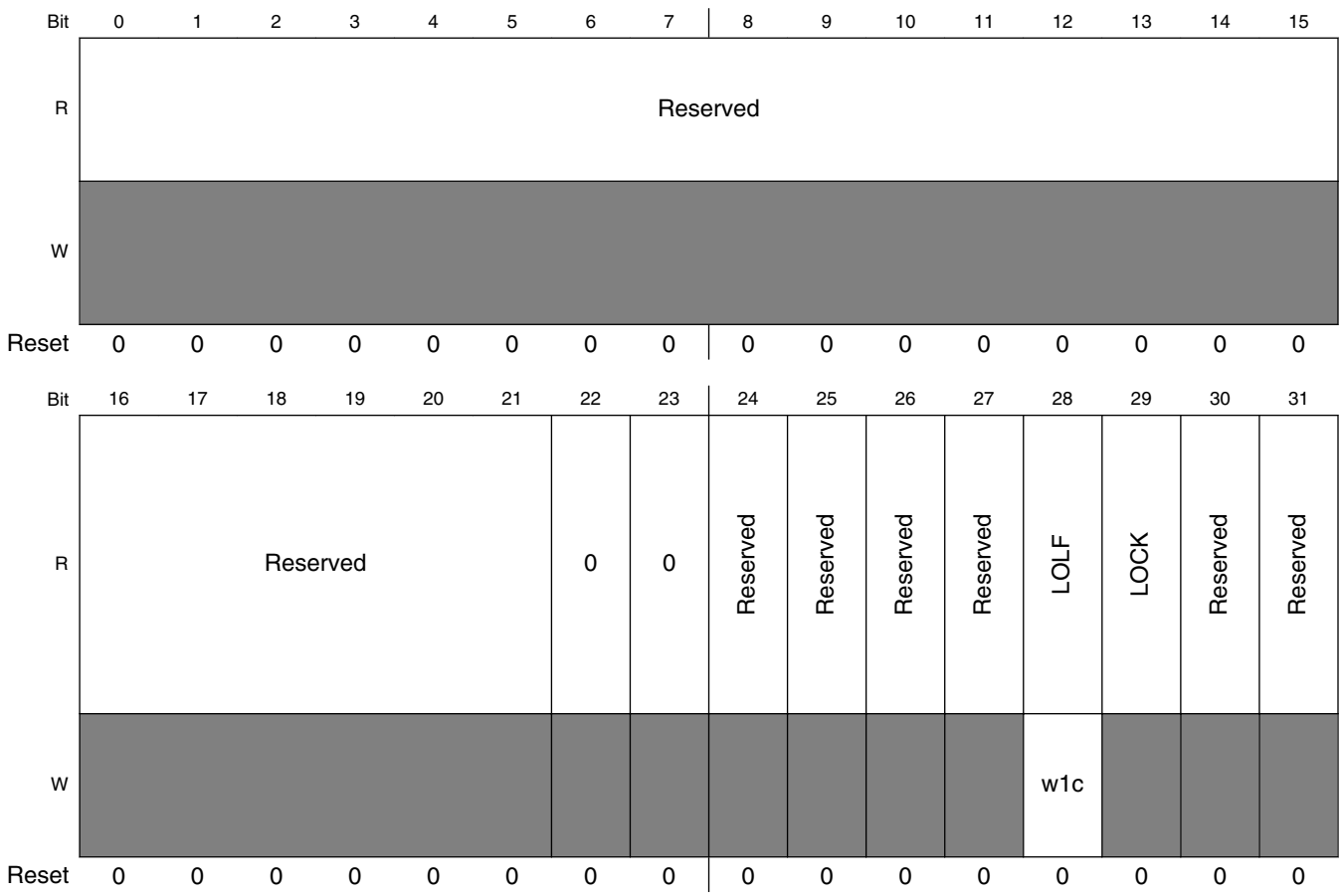
PLLDIG_PLL1CR field descriptions (continued)

Field	Description
	This field indicates the operational state of the PLL. When the PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1. **NOTE:** CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 Bypass mode with PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode, therefore LOLIE bit has no effect in bypass mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

32.5.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)

PLL1SR contains status flags showing the current state of the PLL.

Address: 0h base + 24h offset = 24h



PLLDIG_PLL1SR field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLL1SR field descriptions (continued)

Field	Description
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. LOLF is not asserted if the loss-of-lock condition was detected while the PLL is in bypass mode, but changing from normal to bypass mode will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. If operating in bypass mode, the LOCK bit will still operate normally (for example, assert when the PLL acquires lock or negate when it loses lock). 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

32.5.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)

The PLL1DV register provides the PHI output clock reduced frequency divider (RFDPHI) and loop divider (MFD). If the PLL1DV fields are modified without powering down the PLL, the PLL will lose lock and generate either a reset or interrupt based on which is enabled.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										RFDPHI						Reserved					Reserved				MFD							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*

* Notes:

- MFD field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

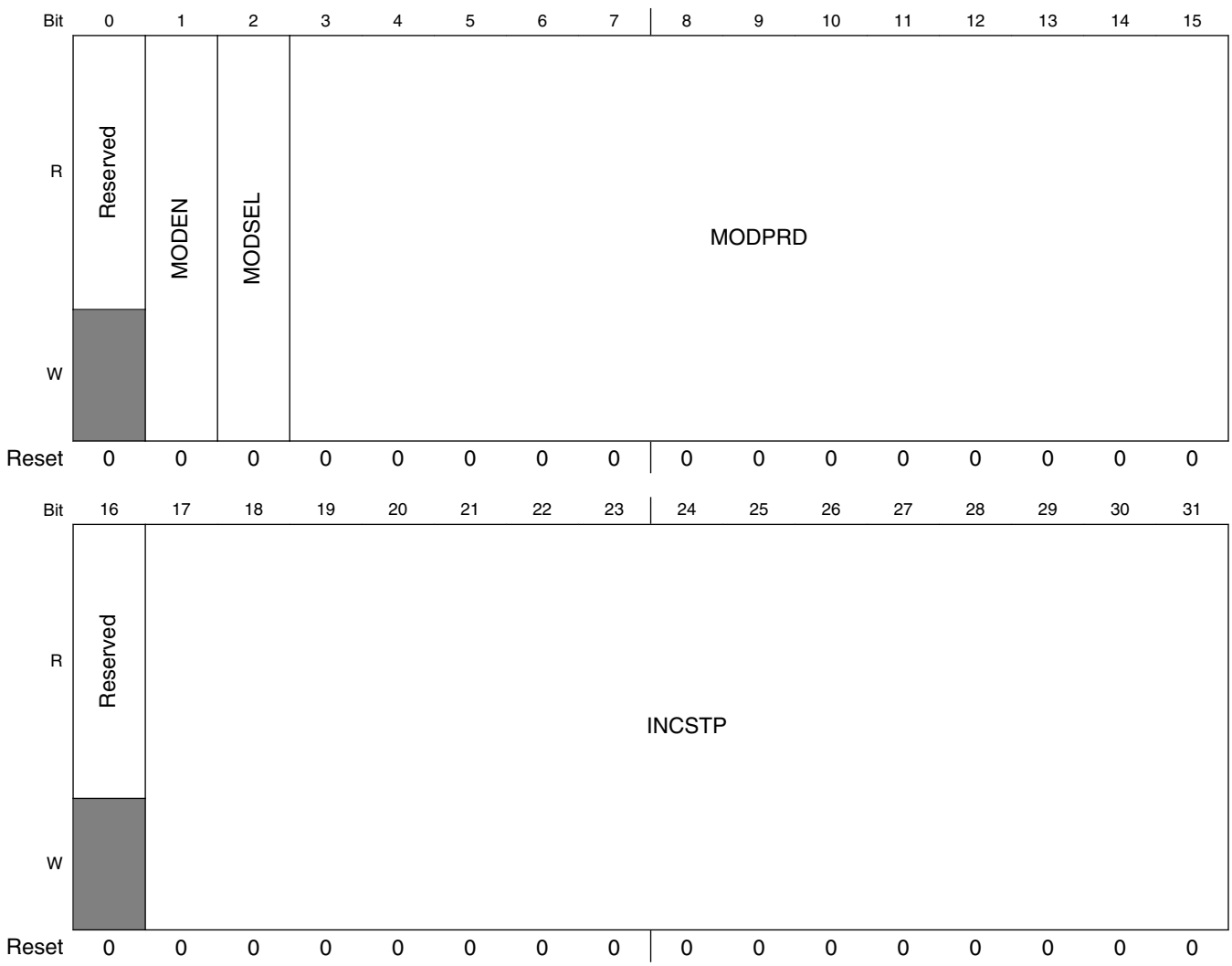
PLLDIG_PLL1DV field descriptions

Field	Description
0–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16–20 Reserved	This field is reserved.
21–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 16...34 (10h...22h). All other values are reserved.

32.5.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)

The PLL1FM register controls enables frequency modulation, and provides controls for modulation mode selection, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). The PLL would then be required to be power cycled to regain normal functionality.

Address: 0h base + 2Ch offset = 2Ch



PLLDIG_PLL1FM field descriptions

Field	Description
0 Reserved	This field is reserved.

Table continues on the next page...

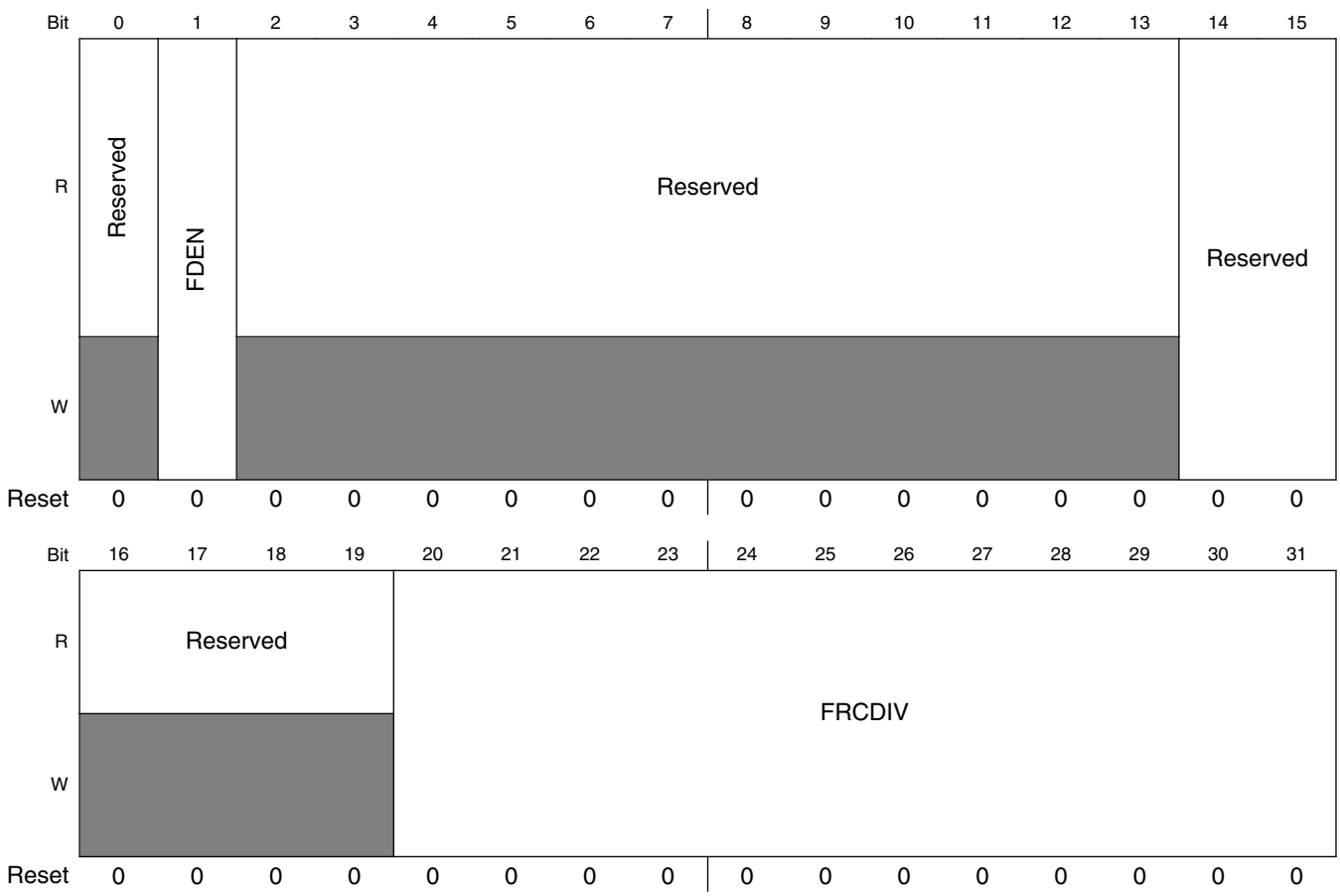
PLLDIG_PLL1FM field descriptions (continued)

Field	Description
1 MODEN	Modulation enable. This bit enables the frequency modulation. 0 Frequency modulation disabled. 1 Frequency modulation enabled.
2 MODSEL	Modulation selection. This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Centre Spread Modulation - Modulation centred around nominal frequency. 1 Down Spread Modulation - Modulation spread below nominal frequency.
3–15 MODPRD	Modulation period. MODPRD is the modulation period variable derived from the formulas shown in Frequency modulation.
16 Reserved	This field is reserved.
17–31 INCSTP	Increment step. This field is the INCSTP variable derived from the formulas shown in Frequency modulation.

32.5.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock. The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 30h offset = 30h



PLLDIG_PLL1FD field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1.

Table continues on the next page...

PLLDIG_PLL1FD field descriptions (continued)

Field	Description
	0 Fractional divide disabled. 1 Fractional divide enabled.
2–13 Reserved	This field is reserved.
14–15 Reserved	Reserved. This field is reserved. **Important:** This field must be written 00b.
16–19 Reserved	This field is reserved.
20–31 FRCDIV	Fractional divide input. When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by the equation in the section Clock configuration.

32.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

32.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for the PLLs are discussed in detail in the *Data Sheet*.

32.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{pll0\_phi} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]} \right)$$

Equation 1. PLL0 PHI Output Frequency

$$f_{pll0\_phi1} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]} \right)$$

Equation 2. PLL0 PHI1 Output Frequency

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 3. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD]}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 4. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{pll0\_VCO} = f_{pll0\_ref} \times \left(\frac{PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]} \right)$$

Equation 5. PLL0 VCO Frequency

$$f_{pll1\_VCO} = f_{pll1\_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}} \right)$$

Equation 6. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1\_VCO} = f_{pll1\_ref} \times PLL1DV[MFD]$$

Equation 7. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

NOTE

f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of the PLL*n*DV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry.

Note

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Initialization information.

Coming out of reset, PLL1 and PLL0 are off and in bypass mode. The recommended procedure to program the PLLs and engage normal mode is described in the 'Clocking' chapter.

32.6.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 32-1. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

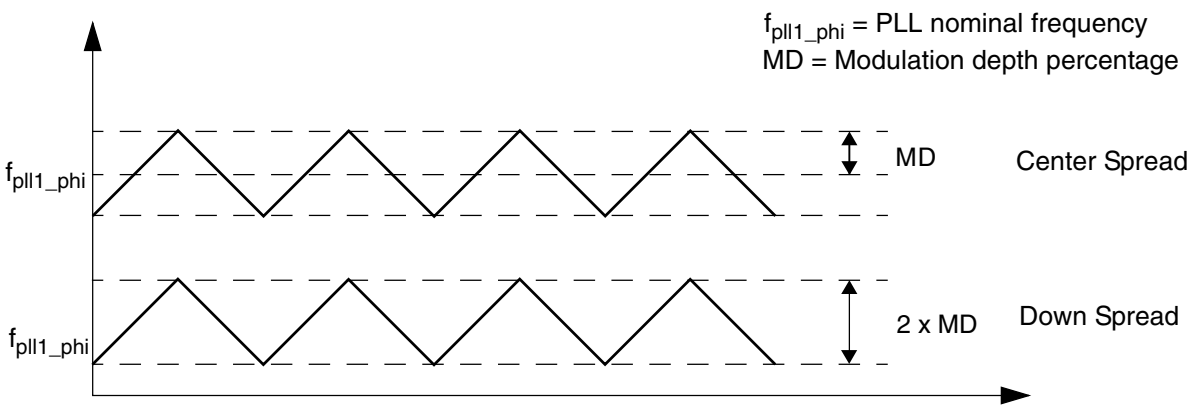


Figure 32-1. Triangular frequency modulation

NOTE

The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys} \times (1 + (MD\% / 100))$.

The following equations define how to calculate PLL1FM[MODPRD] and PLL1FM[INCSTP] based on the output frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1\_ref}}{4 \times f_{mod}}\right)$$

Equation 8. PLL1FM[MODPRD] Calculation

The equation to determine PLL1FM[INCSTP] is shown in Equation 9 on page 1143.

$$PLL1FM[INCSTP] = round\left(\frac{(2^{15}-1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

Equation 9. PLL1FM[INCSTP] Calculation

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Equation 10. PLL1FM[MODPRD] and PLL1FM[INCSTP] Restriction

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is shown in Equation 11 on page 1143.

$$ModulationDepth = round\left(\frac{PLL1FM[MODPRD] \times PLLIFM[INCSTP] \times 100 \times 5}{(2^{15} - 1) \times PLL1DV[MFD]}\right)$$

Equation 11. Modulation Depth

FM parameters should only be changed, and FM enabled, after PLL0 has obtained lock. The sequence for reprogramming the FM is:

1. Power down PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 0, followed by a mode change.
2. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].
3. Program the PLL1FM while PLL1 is powered down.
4. Power up the PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 1, followed by a mode change.
5. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].

32.7 Maximum lock time

This describes the maximum lock time of the PLL across process, supply voltages, and junction temperature. Maximum lock time is for the condition when the operating mode of the PLL has changed from power-down to Normal mode, input control bits are stable, and supplies (analog and digital) have attained levels shown in the *Data Sheet*. It is also the time in which the PLLs regain lock after a loss of lock event, assuming all inputs are stable and supplies are within specifications.

32.8 Initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The Dual PLLDIG initialization procedure is described in the "Clocking" chapter.

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

Chapter 33
Clock Monitor Unit (CMU)

33.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock source CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

33.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN ÷ 4.
- Event generation for various failures detected inside monitoring unit.

33.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

33.3 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 33-1. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

33.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

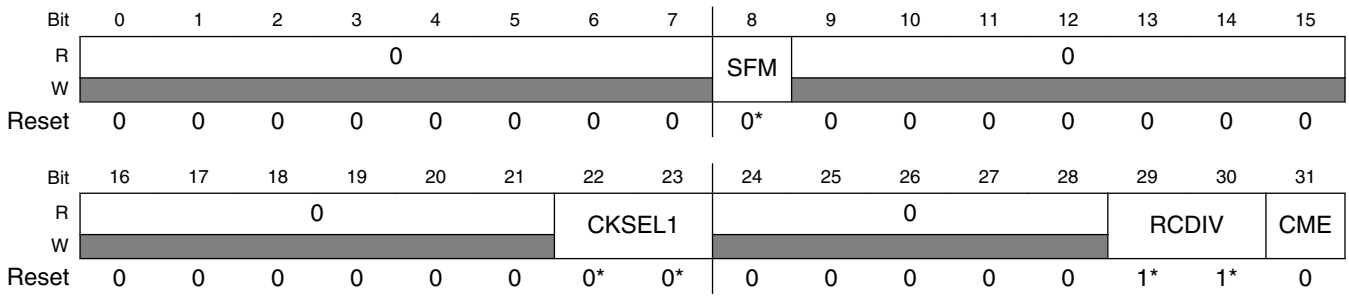
See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	33.4.1/1148
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	33.4.2/1149
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	33.4.3/1150
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	33.4.4/1150
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	33.4.5/1151
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	33.4.6/1153

33.4.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h



* Notes:
- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. **NOTE:** MDR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
29–30 RCDIV	CLKMT0_RMN division factor. These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0\_RMN} \div 2^{CMU\_CSR[RCDIV]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. **NOTE:** Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV]. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4 11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

33.4.2 CMU Frequency Display Register (CMU_FDR)

The CMU_FDR is used to determine the measured frequency of:

- CLKMT0_RMN

with respect to the reference clock CLKMN0_RMT.

NOTE

The CMU_FDR should be read only when CMU_CSR[SFM] = 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0\_RMT} \times CMU\_MDR[MD]) \div CMU\_FDR[FD]$

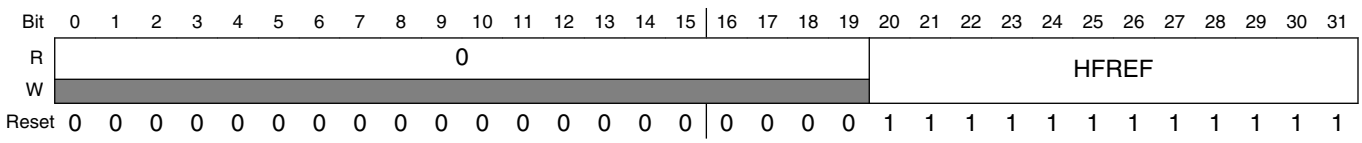
33.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_HFREFR register.

NOTE

For correct synchronization of CMU_HFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + 8h offset = 8h



CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

33.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_LFREFR register.

NOTE

For correct synchronization of CMU_LFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + Ch offset = Ch

Bit	0–19	20–31
R	0	LFREF
W		
Reset	0	0

CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

33.4.5 CMU Interrupt Status Register (CMU_ISR)

NOTE

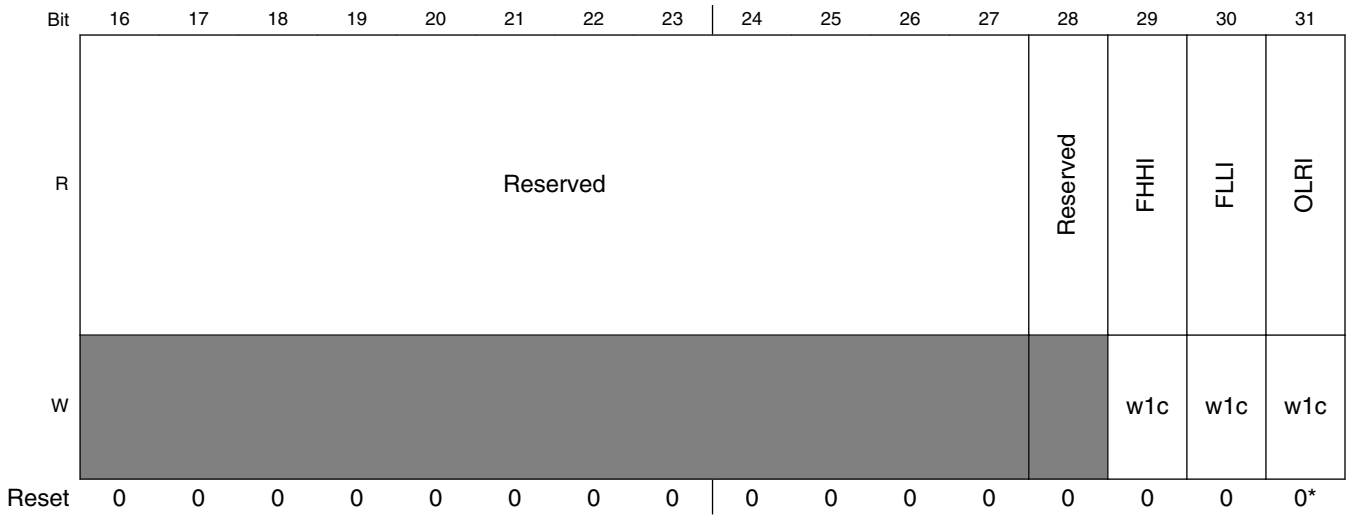
All flags in the CMU_ISR are set asynchronously. This register must be read only after a CMU event is triggered. Otherwise, a read access on this register may fetch an incorrect value (see "Clocking" chapter for interrupt operation) .

NOTE

Before entering low-power stop modes, all clock frequency measurements in the CMU should be disabled. Failing to do so may result in ISR flags being spuriously set.

Address: 0h base + 10h offset = 10h

Bit	0–7	8–15
R	Reserved	
W		
Reset	0	0



* Notes:

- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. **NOTE:** This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled .
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than the high frequency reference value determined by CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than the low frequency reference value determined by CMU_LFREFR[LFREF], and CLKMN1 is 'ON' as signaled by the MC_ME. Software clears this field by writing a '1'. **NOTE:** This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled . 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$event status. This bit is set by hardware when the f_{CLKMN0_RMT} is less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. **NOTE:** While entering STOP mode, OLRI may be triggered due to stoppage of XOSC. To avoid this, disable the XOSC before entering STOP mode.

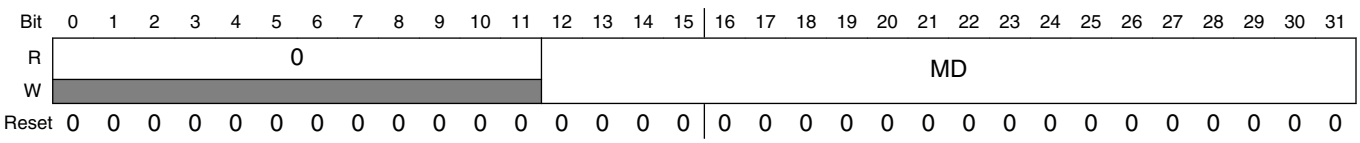
Table continues on the next page...

CMU_ISR field descriptions (continued)

Field	Description
	0 No OLR event 1 OLR event occurred

33.4.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h



CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1. **NOTE:** Only values that do not cause a CMU_FDR[FD] overflow shall be written to CMU_MDR[MD]. There is no internal status of a CMU_FDR[FD] overflow.

33.5 Functional description

33.5.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is

complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR[FD] as shown in the following equation:

$$f_{sel} = \left(f_{CLKMNO\_RMT} \times MDR[MD]\right) / \ FDR[FD]$$

33.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN ÷ $2^{CMU\_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

f_{CLKMN0_RMT} must be greater than $f_{CLKMT0\_RMN} \div 2^{CMU\_CSR[RCDIV]}$ by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

33.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

An example of determining the $HFREF_{Actual}$ is as follows. Assume a f_{CLKMT0_RMN} = 16 MHz with a accuracy of +/-5% (see *Data Sheet* for actual value). In order to monitor a f_{CLKMN1} = 200 MHz, the ideal $HFREF_{Ideal}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($HFREF_{Actual}$ = ($HFREF_{Ideal}$ ÷ 0.95). The actual LFREF value will be 762 when accuracy is taken into consideration ($LFREF_{Actual}$ = $LFREF_{Ideal}$ ÷ 1.05).

Chapter 34
Clock Generation Module (MC_CGM)

34.1 Introduction

34.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

The following figure shows the MC_CGM block diagram.

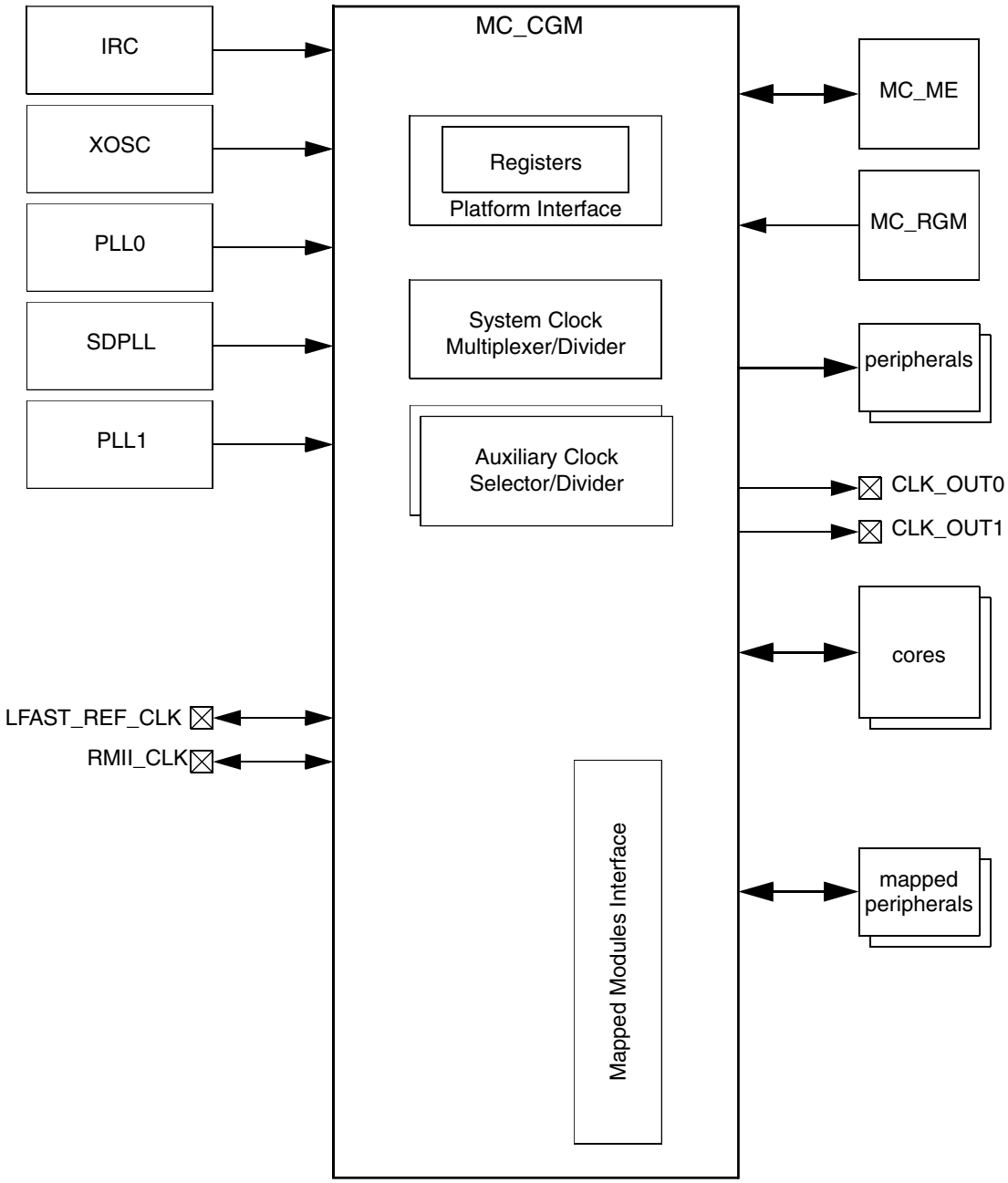


Figure 34-1. MC_CGM block diagram

34.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- performs progressive system clock frequency change depending on MC_ME mode configuration

- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

34.2 External signal description

The MC_CGM delivers output clocks to the CLK_OUT0 and CLK_OUT1 pins for off-chip use and/or observation.

The LFAST_Ref pin can be either driven by the MC_CGM or used as an input to source the LFAST_CLK.

The RMII_CLK pin can be either driven by the MC_CGM or used as an input to source the ENET_CLK and ENET_TIME_CLK.

34.3 Memory mapped registers

The bytes are ordered according to big endian. For example, the **CGM_PCS_DIVC1** register **RATE** bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

NOTE

Any access to unused registers as well as write accesses to read-only registers will

- not change register content
- cause a transfer error

Any access to offsets 0x80Ch, 0x82Ch, 0x9E8h will not return transfer error.

NOTE

Do not write on reserved fields of registers

NOTE

The following registers are also writable in user mode

- CGM_PCS_DIVSn
- CGM_PCS_DIVEn

- CGM_PCS_DIVCn
- CGM_PCS_SDUR

NOTE

Accessing following registers in user mode does not have any effect

- CGM_SC_DIV_RC
- CGM_DIV_UPD_TYPE
- CGM_DIV_UPD_TRIG
- CGM_DIV_UPD_STAT

NOTE

The following registers are reset to their default values during a POR event only:

- CGM_PCS_SDUR
- CGM_PCS_DIVC1
- CGM_PCS_DIVS1
- CGM_PCS_DIVE1
- CGM_PCS_DIVC2
- CGM_PCS_DIVS2
- CGM_PCS_DIVE2
- CGM_PCS_DIVC4
- CGM_PCS_DIVS4
- CGM_PCS_DIVE4
- CGM_AC0_SC
- CGM_AC1_SC
- CGM_AC2_SC
- CGM_AC3_SC
- CGM_AC4_SC
- CGM_AC7_SC
- CGM_AC8_SC
- CGM_AC10_SC
- CGM_AC11_SC
- CGM_AC12_SC
- CGM_AC13_SC

CGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
700	PCS Switch Duration Register (CGM_PCS_SDUR)	8	R/W	00h	34.3.1/1163
704	PCS Divider Change Register 1 (CGM_PCS_DIVC1)	32	R/W	03E7_0000h	34.3.2/1164
708	PCS Divider End Register 1 (CGM_PCS_DIVE1)	32	R/W	0000_03E7h	34.3.3/1164

Table continues on the next page...

CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
70C	PCS Divider Start Register 1 (CGM_PCS_DIVS1)	32	R/W	0000_03E7h	34.3.4/1165
710	PCS Divider Change Register 2 (CGM_PCS_DIVC2)	32	R/W	03E7_0000h	34.3.5/1166
714	PCS Divider End Register 2 (CGM_PCS_DIVE2)	32	R/W	0000_03E7h	34.3.6/1167
718	PCS Divider Start Register 2 (CGM_PCS_DIVS2)	32	R/W	0000_03E7h	34.3.7/1167
728	PCS Divider Change Register 4 (CGM_PCS_DIVC4)	32	R/W	03E7_0000h	34.3.8/1168
72C	PCS Divider End Register 4 (CGM_PCS_DIVE4)	32	R/W	0000_03E7h	34.3.9/1169
730	PCS Divider Start Register 4 (CGM_PCS_DIVS4)	32	R/W	0000_03E7h	34.3.10/ 1169
7D0	System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)	32	R/W	0000_0001h	34.3.11/ 1170
7D4	Divider Update Type Register (CGM_DIV_UPD_TYPE)	32	R/W	0000_0000h	34.3.12/ 1171
7D8	Divider Update Trigger Register (CGM_DIV_UPD_TRIG)	32	R/W	0000_0000h	34.3.13/ 1172
7DC	Divider Update Status Register (CGM_DIV_UPD_STAT)	32	R/W	0000_0000h	34.3.14/ 1173
7E4	System Clock Select Status Register (CGM_SC_SS)	32	R	0008_0000h	34.3.15/ 1175
7E8	System Clock Divider 0 Configuration Register (CGM_SC_DC0)	32	R/W	8001_0000h	34.3.16/ 1177
7EC	System Clock Divider 1 Configuration Register (CGM_SC_DC1)	32	R/W	8000_0000h	34.3.17/ 1178
7F0	System Clock Divider 2 Configuration Register (CGM_SC_DC2)	32	R/W	8000_0000h	34.3.18/ 1179
7F4	System Clock Divider 3 Configuration Register (CGM_SC_DC3)	32	R/W	8003_0000h	34.3.19/ 1180
7F8	System Clock Divider 4 Configuration Register (CGM_SC_DC4)	32	R/W	8001_0000h	34.3.20/ 1181
7FC	System Clock Divider 5 Configuration Register (CGM_SC_DC5)	32	R/W	8001_0000h	34.3.21/ 1182
800	Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)	32	R/W	0000_0000h	34.3.22/ 1183
804	Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)	32	R	0000_0000h	34.3.23/ 1183
808	Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)	32	R/W	8000_0000h	34.3.24/ 1184
810	Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)	32	R/W	8000_0000h	34.3.25/ 1185
820	Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)	32	R/W	0000_0000h	34.3.26/ 1186
824	Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)	32	R	0000_0000h	34.3.27/ 1187
828	Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)	32	R/W	8000_0000h	34.3.28/ 1188

Table continues on the next page...

CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
840	Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)	32	R/W	0000_0000h	34.3.29/ 1189
844	Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)	32	R	0000_0000h	34.3.30/ 1189
848	Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)	32	R/W	8000_0000h	34.3.31/ 1190
860	Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)	32	R/W	0000_0000h	34.3.32/ 1191
864	Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)	32	R	0000_0000h	34.3.33/ 1192
880	Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)	32	R/W	0000_0000h	34.3.34/ 1193
884	Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)	32	R	0000_0000h	34.3.35/ 1194
8E0	Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)	32	R/W	0000_0000h	34.3.36/ 1195
8E4	Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)	32	R	0000_0000h	34.3.37/ 1195
8E8	Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)	32	R/W	8000_0000h	34.3.38/ 1196
900	Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)	32	R/W	0000_0000h	34.3.39/ 1197
904	Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)	32	R	0000_0000h	34.3.40/ 1198
908	Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)	32	R/W	8000_0000h	34.3.41/ 1199
928	Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)	32	R/W	8000_0000h	34.3.42/ 1200
940	Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)	32	R/W	0000_0000h	34.3.43/ 1201
944	Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)	32	R	0000_0000h	34.3.44/ 1201
948	Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)	32	R/W	8000_0000h	34.3.45/ 1202
960	Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)	32	R/W	0000_0000h	34.3.46/ 1203
964	Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)	32	R	0000_0000h	34.3.47/ 1204
968	Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)	32	R/W	8000_0000h	34.3.48/ 1205
980	Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)	32	R/W	0000_0000h	34.3.49/ 1206
984	Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)	32	R	0000_0000h	34.3.50/ 1206

Table continues on the next page...

CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
988	Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)	32	R/W	8000_0000h	34.3.51/ 1207
9A0	Auxiliary Clock 13 Select Control Register (CGM_AC13_SC)	32	R/W	0000_0000h	34.3.52/ 1208
9A4	Auxiliary Clock 13 Select Status Register (CGM_AC13_SS)	32	R/W	0000_0000h	34.3.53/ 1209
9A8	Auxiliary Clock 13 Divider 0 Configuration Register (CGM_AC13_DC0)	32	R/W	8000_0000h	34.3.54/ 1210
9C8	Auxiliary Clock 14 Divider 0 Configuration Register (CGM_AC14_DC0)	32	R/W	8000_0000h	34.3.55/ 1211

34.3.1 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See Progressive System Clock Switching for details on how to set this value.

Address: 0h base + 700h offset = 700h

Bit	0	1	2	3	4	5	6	7
Read Write	SDUR							
Reset	0	0	0	0	0	0	0	0

CGM_PCS_SDUR field descriptions

Field	Description
0–7 SDUR	**Switch Duration** — This value defines the duration of one PCS clock switch step in terms of IRC_CLK cycles.

34.3.2 PCS Divider Change Register 1 (CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the XOSC_CLK on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 704h offset = 704h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	**Divider Change Initial Value** — This is initial change value of the clock divider for the clock ramp-up phase when switching to the XOSC_CLK.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	**Divider Change Rate** — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the XOSC_CLK.

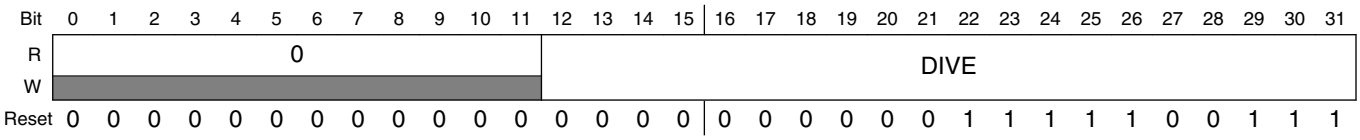
34.3.3 PCS Divider End Register 1 (CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the XOSC_CLK on ramp-down. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 708h offset = 708h



CGM_PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	**Divider End Value** — This is the clock divider end value for the clock ramp-down phase when switching from the XOSC_CLK.

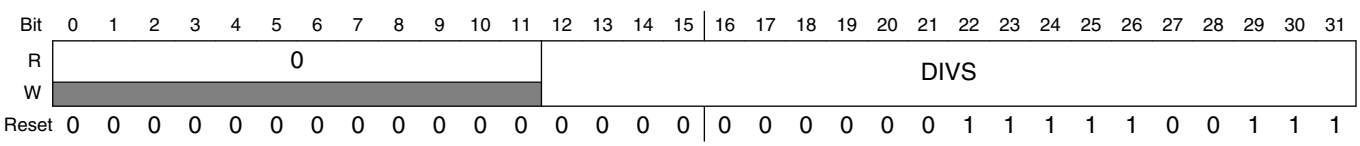
34.3.4 PCS Divider Start Register 1 (CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the XOSC_CLK on ramp-up. Register *n* corresponds to system clock source *n*. See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 70Ch offset = 70Ch



CGM_PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	**Divider Start Value** — This is the start value of the clock divider for the clock ramp-up phase when switching to the XOSC_CLK.

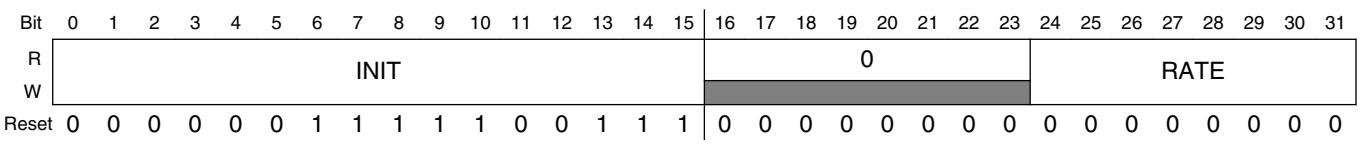
34.3.5 PCS Divider Change Register 2 (CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0_PHI_CLK on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 710h offset = 710h



CGM_PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	**Divider Change Initial Value** — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0_PHI_CLK.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	**Divider Change Rate** — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0_PHI_CLK.

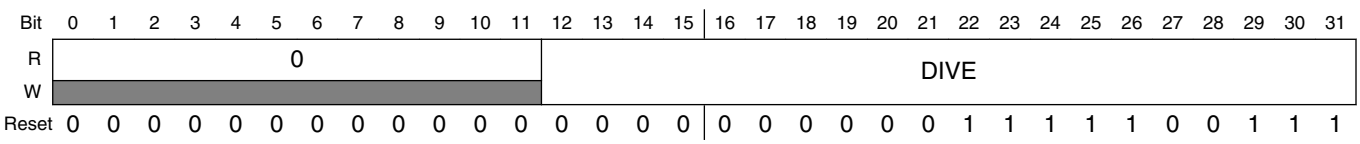
34.3.6 PCS Divider End Register 2 (CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0_PHI_CLK on ramp-down. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 714h offset = 714h



CGM_PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	**Divider End Value** — This is the clock divider end value for the clock ramp-down phase when switching from the PLL0_PHI_CLK.

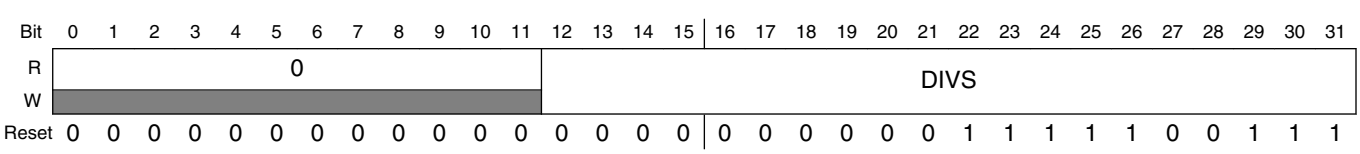
34.3.7 PCS Divider Start Register 2 (CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0_PHI_CLK on ramp-up. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 718h offset = 718h



CGM_PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	**Divider Start Value** — This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0_PHI_CLK.

34.3.8 PCS Divider Change Register 4 (CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1_PHI_CLK on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 728h offset = 728h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_PCS_DIVC4 field descriptions

Field	Description
0–15 INIT	**Divider Change Initial Value** — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1_PHI_CLK.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	**Divider Change Rate** — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1_PHI_CLK.

34.3.9 PCS Divider End Register 4 (CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1_PHI_CLK on ramp-down. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 72Ch offset = 72Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

CGM_PCS_DIVE4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	**Divider End Value** — This is the clock divider end value for the clock ramp-down phase when switching from the PLL1_PHI_CLK.

34.3.10 PCS Divider Start Register 4 (CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1_PHI_CLK on ramp-up. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 730h offset = 730h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

CGM_PCS_DIVS4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	**Divider Start Value** — This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1_PHI_CLK.

34.3.11 System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

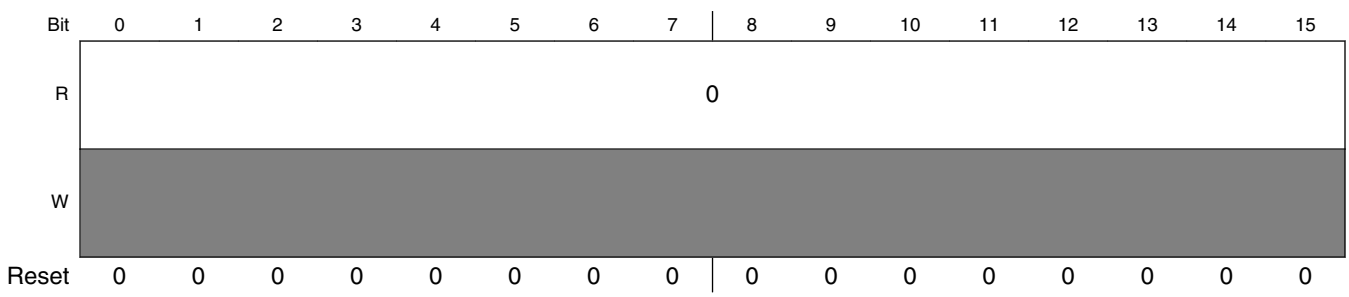
An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations.

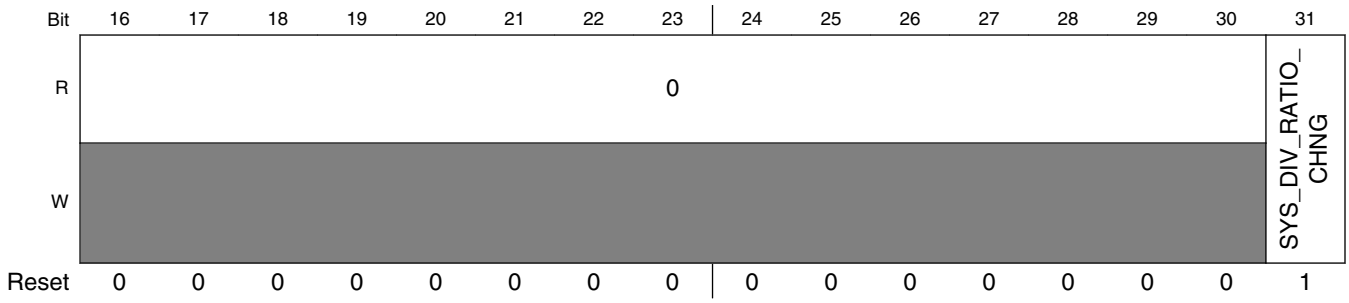
An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3.

The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

Address: 0h base + 7D0h offset = 7D0h





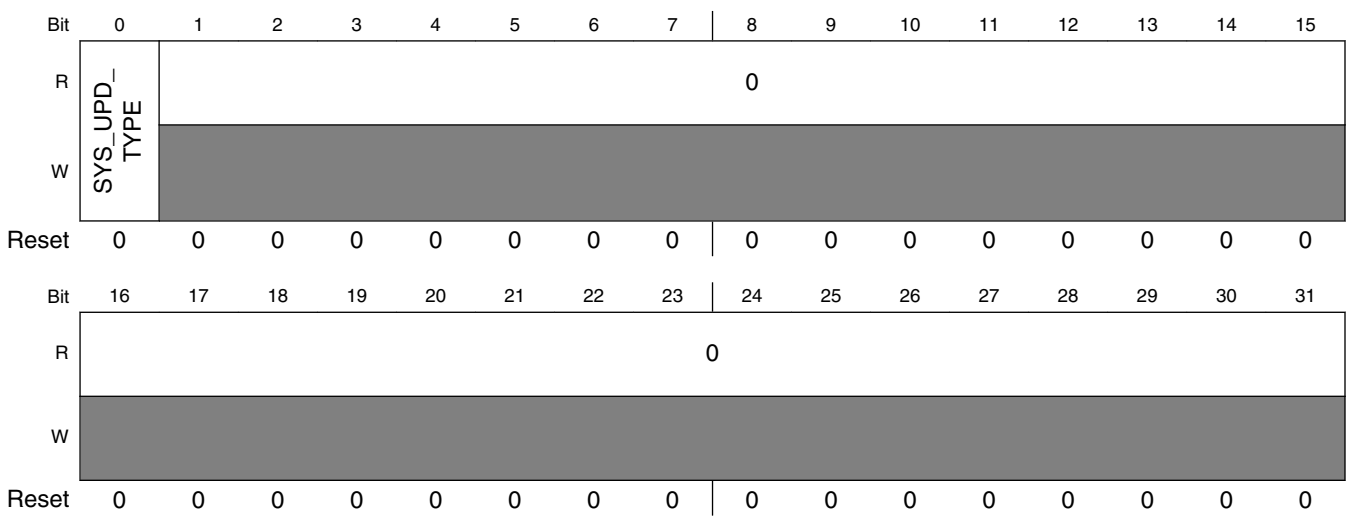
CGM_SC_DIV_RC field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SYS_DIV_RATIO_CHNG	System Clock Divider Ratio Change 0 the system clock divider ratios will not change with the next system clock divider configuration update 1 the system clock divider ratios will change with the next system clock divider configuration update

34.3.12 Divider Update Type Register (CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers wil take effect.

Address: 0h base + 7D4h offset = 7D4h



CGM_DIV_UPD_TYPE field descriptions

Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type

Table continues on the next page...

CGM_DIV_UPD_TYPE field descriptions (continued)

Field	Description
	0 The configuration for each system clock divider is updated immediately on writing to the corersponding CGM_SC_DCi register 1 The configuration for each system clock divider is preloaded on writing to the corersponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register
1–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.13 Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Address: 0h base + 7D8h offset = 7D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	DIV_UPD_TRIGGER																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

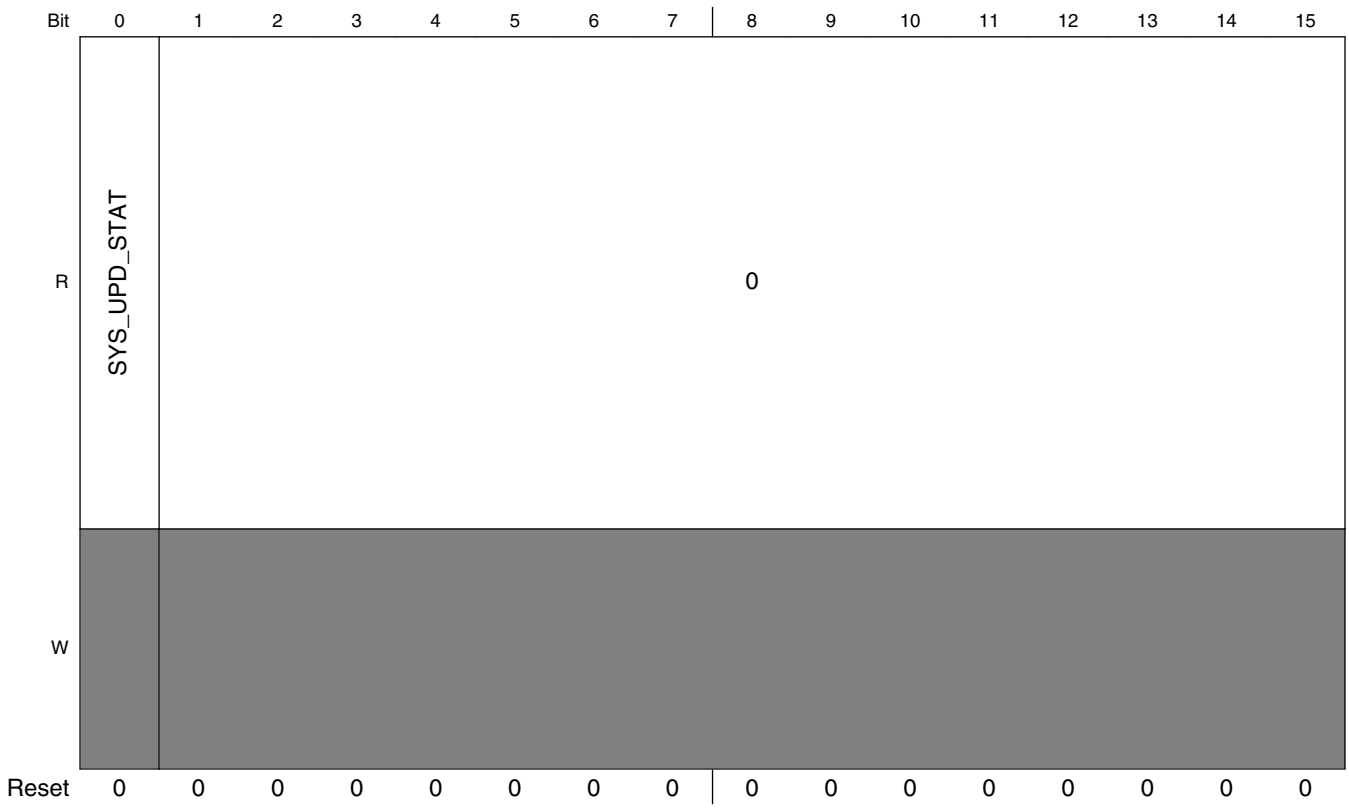
CGM_DIV_UPD_TRIG field descriptions

Field	Description
0–31 DIV_UPD_ TRIGGER	Writing any value to this register will cause all dividers that have corresponding bits set to ‘1’ in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

34.3.14 Divider Update Status Register (CGM_DIV_UPD_STAT)

This register provides status whether the dividers associated with a clock mux are in the process of being updated.

Address: 0h base + 7DCh offset = 7DCh



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	AUX14_UPD_STAT	AUX13_UPD_STAT	AUX12_UPD_STAT	AUX11_UPD_STAT	AUX10_UPD_STAT	AUX9_UPD_STAT	AUX8_UPD_STAT	AUX7_UPD_STAT	0				AUX2_UPD_STAT	AUX1_UPD_STAT	AUX0_UPD_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_DIV_UPD_STAT field descriptions

Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status 0 no system clock divider configuration updates are in process or divider update has finished 1 the configuration for at least one system clock divider is being updated
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 AUX14_UPD_ STAT	Auxillary Clock Divider 14 Update Status 0 no Auxillary divider 14 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 14 clock divider is being updated
18 AUX13_UPD_ STAT	Auxillary Clock Divider 13 Update Status 0 no Auxillary divider 13 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 13 clock divider is being updated
19 AUX12_UPD_ STAT	Auxillary Clock Divider 12 Update Status 0 no Auxillary divider 12 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 12 clock divider is being updated
20 AUX11_UPD_ STAT	Auxillary Clock Divider 11 Update Status 0 no Auxillary divider 11 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 11 clock divider is being updated

Table continues on the next page...

CGM_DIV_UPD_STAT field descriptions (continued)

Field	Description
21 AUX10_UPD_ STAT	Auxillary Clock Divider 10 Update Status 0 no Auxillary divider 10 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 10 clock divider is being updated
22 AUX9_UPD_ STAT	Auxillary Clock Divider 9 Update Status 0 no Auxillary divider 9 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 9 clock divider is being updated
23 AUX8_UPD_ STAT	Auxillary Clock Divider 8 Update Status 0 no Auxillary divider 8 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 8 clock divider is being updated
24 AUX7_UPD_ STAT	Auxillary Clock Divider 7 Update Status 0 no Auxillary divider 7 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 7 clock divider is being updated
25–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AUX2_UPD_ STAT	Auxillary Clock Divider 2 Update Status 0 no Auxillary divider 2 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 2 clock divider is being updated
30 AUX1_UPD_ STAT	Auxillary Clock Divider 1 Update Status 0 no Auxillary divider 1 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 1 clock divider is being updated
31 AUX0_UPD_ STAT	Auxillary Clock Divider 0 Update Status 0 no Auxillary divider 0 configuration updates are in process or divider update has finished 1 the configuration for Auxillary 0 clock divider is being updated

34.3.15 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0h base + 7E4h offset = 7E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				SELSTAT				0				SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_SC_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	**System Clock Source Selection Status** — This value indicates the current source for the system clock. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–14 SWTRG	**Switch Trigger cause** — This value indicates the cause for the latest clock source switch. 000 reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to IRC_CLK due to SAFE mode request or reset succeeded 101 switch to IRC_CLK due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved 111 reserved **NOTE:** The reset value of SWTRG may reflect 001 instead of 100, if offline Self test has been executed.
15 SWIP	**Switch In Progress** 0 clock source switching has completed 1 clock source switching is in progress
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

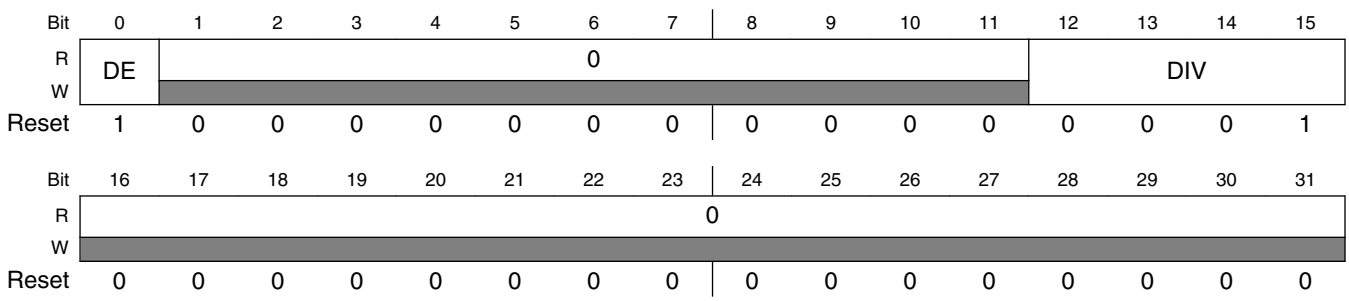
34.3.16 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7E8h offset = 7E8h



CGM_SC_DC0 field descriptions

Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider 1 Division Value** — The resultant SYS_CLK will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the SYS_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.17 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7ECh offset = 7ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_SC_DC1 field descriptions

Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider 1 Division Value** — The resultant CORE1_CLK will have a period ’DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the CORE1_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

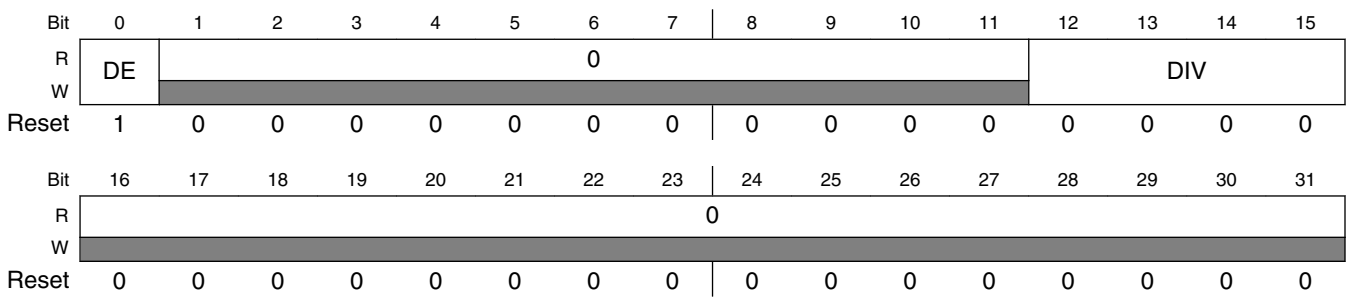
34.3.18 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register controls system clock divider 2. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F0h offset = 7F0h



CGM_SC_DC2 field descriptions

Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider 1 Division Value** — The resultant CORE2_CLK will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the CORE2_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.19 System Clock Divider 3 Configuration Register (CGM_SC_DC3)

This register controls system clock divider 3. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F4h offset = 7F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_SC_DC3 field descriptions

Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider 1 Division Value** — The resultant PBRIDGE_n_CLK will have a period ’DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the PBRIDGE_n_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.20 System Clock Divider 4 Configuration Register (CGM_SC_DC4)

This register controls system clock divider 4. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7F8h offset = 7F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_SC_DC4 field descriptions

Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider 1 Division Value** — The resultant DMA_CLK will have a period ’DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the DMA_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.21 System Clock Divider 5 Configuration Register (CGM_SC_DC5)

This register controls system clock divider 5. See System Clock Divider Synchronization for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 7FCh offset = 7FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_SC_DC5 field descriptions

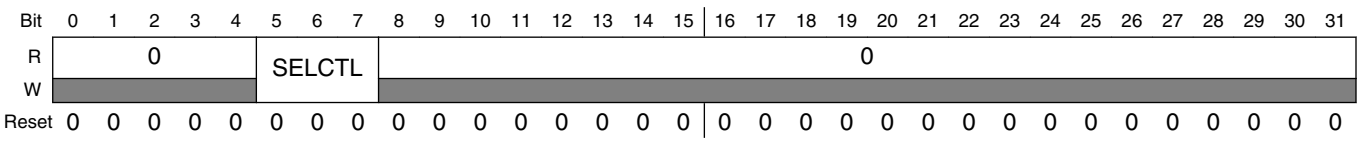
Field	Description
0 DE	**Divider 1 Enable** — This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	**Divider 1 Division Value** — The resultant CORE0_CLK will have a period ’DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the CORE0_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.22 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- divided by auxiliary clock 0 divider 0: MC_CLK
- divided by auxiliary clock 0 divider 2: ADC_CLK

Address: 0h base + 800h offset = 800h



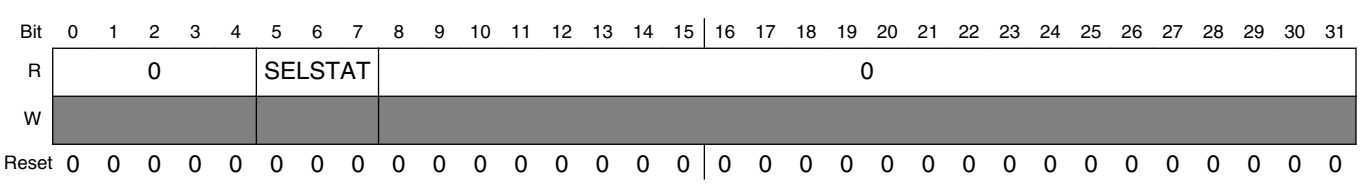
CGM_AC0_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 0 Source Selection Control** — Selects the source for auxiliary clock 0. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.23 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0h base + 804h offset = 804h



CGM_AC0_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 0 Source Selection Status** — This value indicates the current source for auxiliary clock 0. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.24 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 808h offset = 808h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC0_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider Division Value** — The resultant MC_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the MC_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.25 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 810h offset = 810h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC0_DC2 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CGM_AC0_DC2 field descriptions (continued)

Field	Description
11–15 DIV	**Divider Division Value** — The resultant ADC_CLK will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.26 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- divided by auxiliary clock 1 divider 0: FR_CLK

Address: 0h base + 820h offset = 820h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC1_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 1 Source Selection Control** — This value selects the current source for auxiliary clock 1. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.27 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 1 source selection.

Address: 0h base + 824h offset = 824h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC1_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 1 Source Selection Status** — This value indicates the current source for auxiliary clock 1. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.28 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 1 divider 0: FR_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 828h offset = 828h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC1_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	**Divider Division Value** — The resultant FR_CLK will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FR_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.29 Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

This register is used to select the current clock source for the following clocks:

- divided by auxiliary clock 2 divider 0: CAN0_CLK

Address: 0h base + 840h offset = 840h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC2_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 2 Source Selection Control** — This value selects the current source for auxiliary clock 2. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.30 Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)

This register provides the current CAN source selection.

Address: 0h base + 844h offset = 844h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC2_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 2 Source Selection Status** — This value indicates the current source for the CAN clock. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.31 Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0: CAN_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 848h offset = 848h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC2_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0

Table continues on the next page...

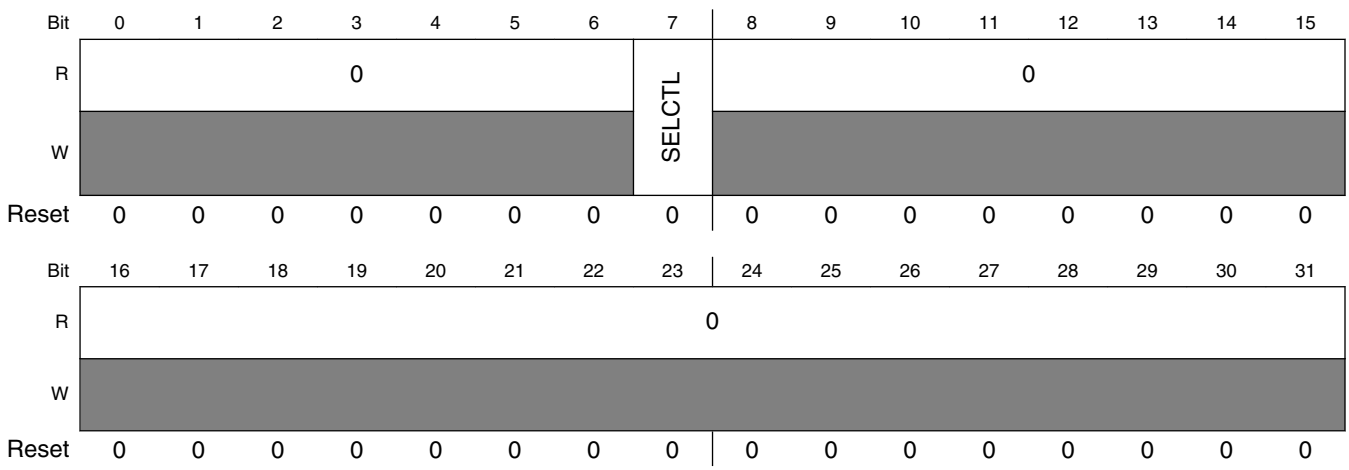
CGM_AC2_DC0 field descriptions (continued)

Field	Description
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	**Divider Division Value** — The resultant CAN0_CLK will have a period 'DIV + 1' times that of the auxiliary clock 2. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the CAN0_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.32 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

Address: 0h base + 860h offset = 860h



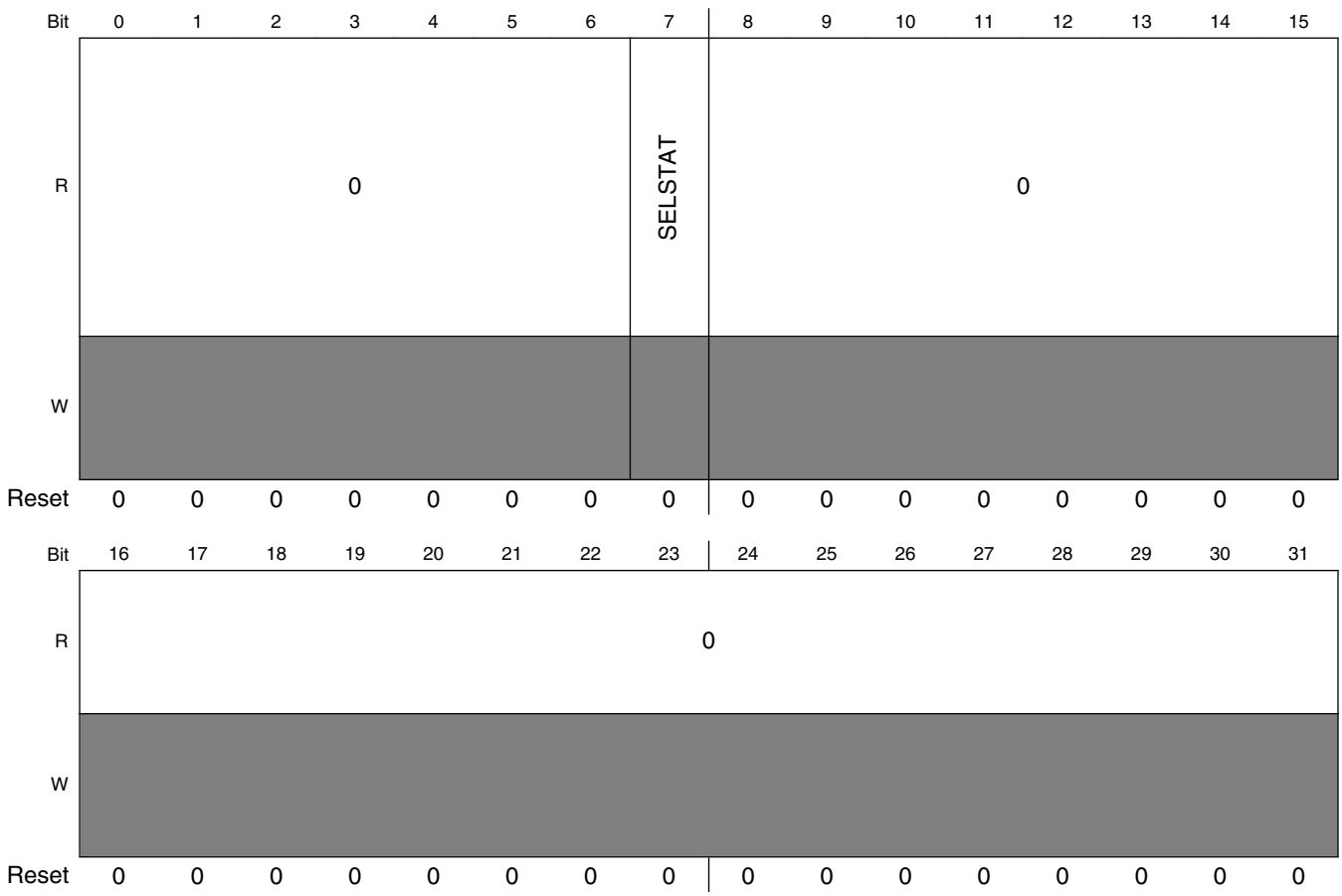
CGM_AC3_SC field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	**Auxiliary Clock 3 Source Selection Control** — This value selects the current source for auxiliary clock 3. 0 IRC_CLK 1 XOSC_CLK
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.33 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Address: 0h base + 864h offset = 864h



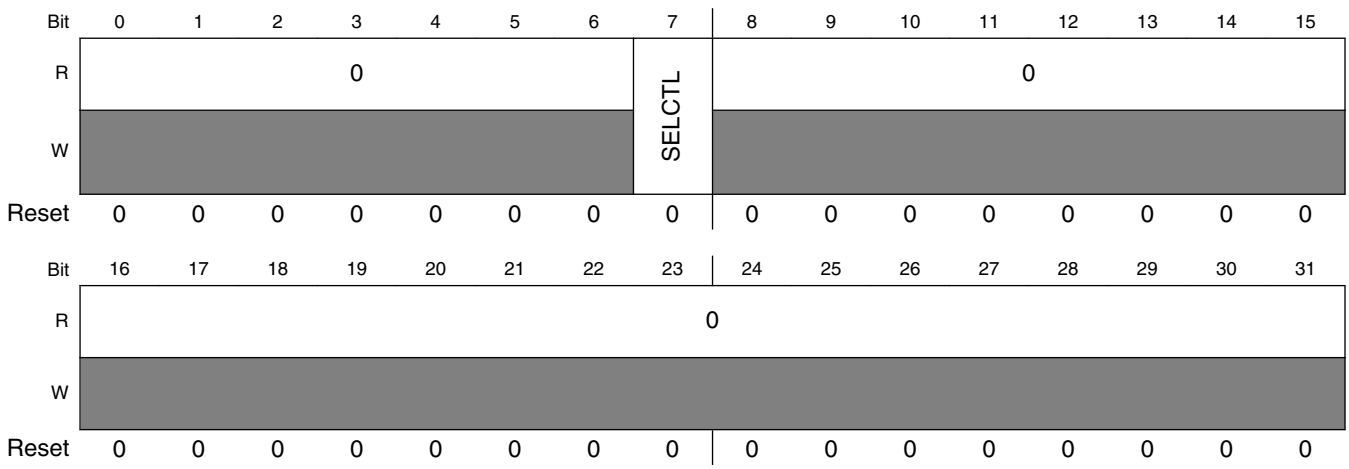
CGM_AC3_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	**Auxiliary Clock 3 Source Selection Status** — This value indicates the current source for auxiliary clock 3. 0 IRC_CLK 1 XOSC_CLK
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.34 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

Address: 0h base + 880h offset = 880h



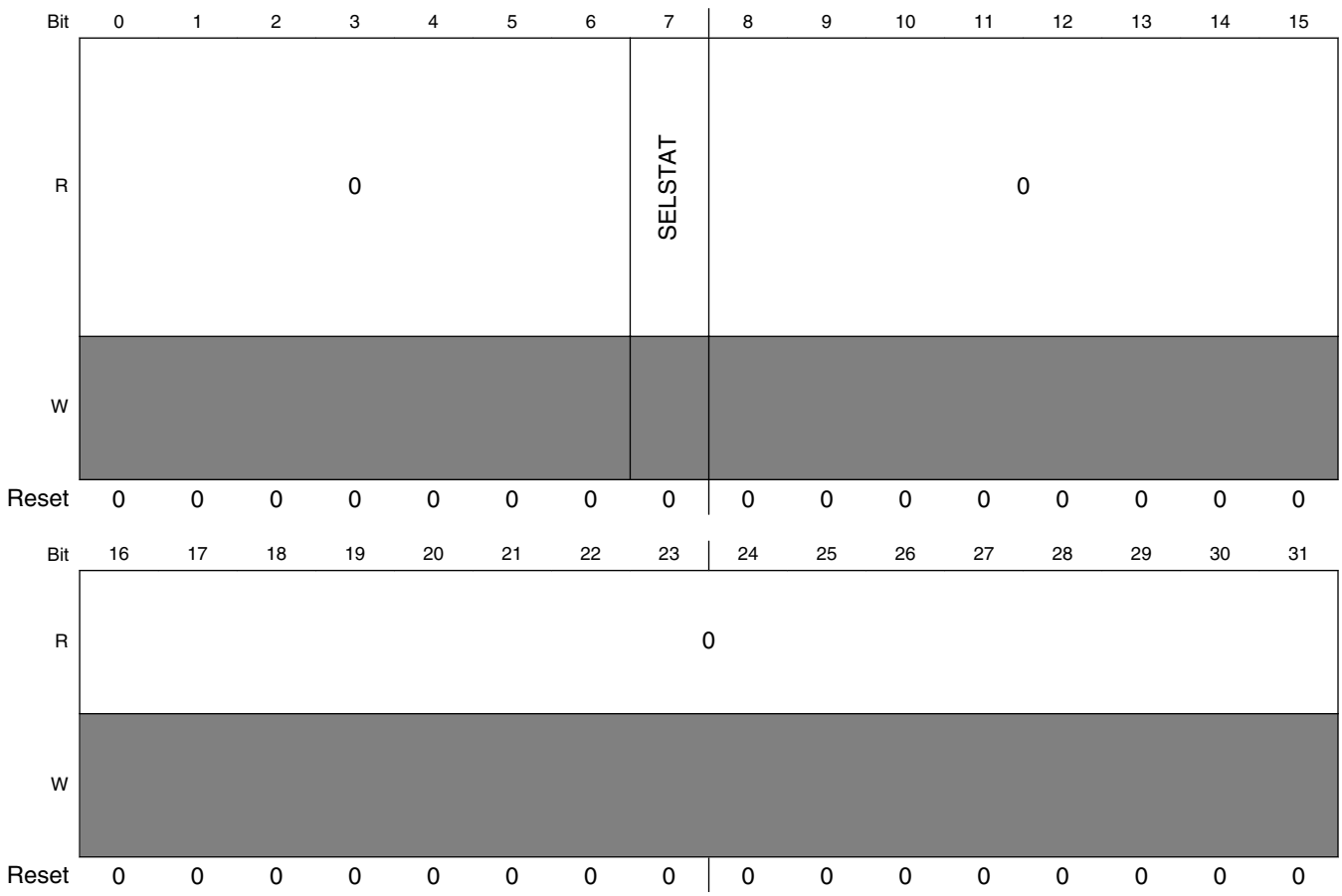
CGM_AC4_SC field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	**Auxiliary Clock 4 Source Selection Control** — This value selects the current source for auxiliary clock 4. 0 PLL0_PHI1_CLK 1 XOSC_CLK
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.35 Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Address: 0h base + 884h offset = 884h



CGM_AC4_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	**Auxiliary Clock 4 Source Selection Status** — This value indicates the current source for auxiliary clock 4. 0 PLL0_PHI1_CLK 1 XOSC_CLK
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.36 Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

This register is used to select the current clock source for the following clock.

- divided by auxiliary clock 7 divider 0: SPT_CLK

Address: 0h base + 8E0h offset = 8E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC7_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 7 Source Selection Control** — This value selects the current source for auxiliary clock 7. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.37 Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

This register provides the current auxiliary clock 7 source selection.

Address: 0h base + 8E4h offset = 8E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC7_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 7 Source Selection Status** — This value indicates the current source for auxiliary clock 7. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.38 Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

This register controls auxiliary clock 7 divider 0: SPT_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8E8h offset = 8E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC7_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 7 divider 0 1 Enable auxiliary clock 7 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider Division Value** — The resultant SPT_CLK will have a period 'DIV + 1' times that of the auxiliary clock 7. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the SPT_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.39 Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

This register is used to select the current clock source for the following clock.

- divided by auxiliary clock 8 divider 0: LFAST_CLK.

Address: 0h base + 900h offset = 900h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

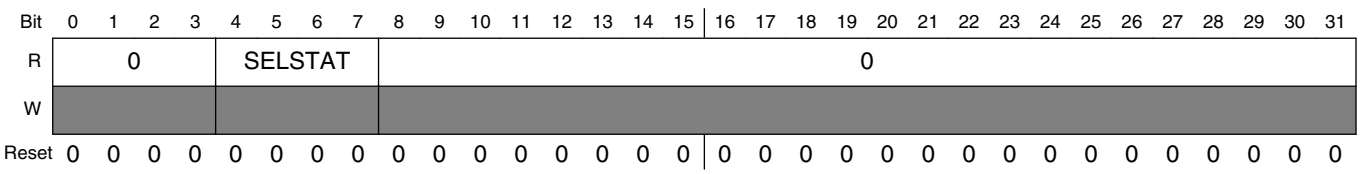
CGM_AC8_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 8 Source Selection Control** — This value selects the current source for auxiliary clock 8. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 LFAST_REF_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.40 Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Address: 0h base + 904h offset = 904h



CGM_AC8_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	**Auxiliary Clock 8 Source Selection Status** — This value indicates the current source for auxiliary clock 8. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 LFAST_REF_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

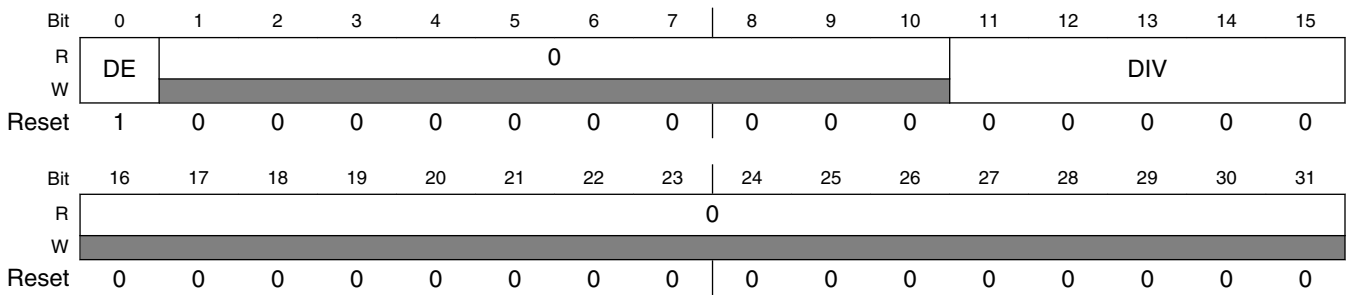
34.3.41 Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0: LFAST_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 908h offset = 908h



CGM_AC8_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	**Divider Division Value** — The resultant LFAST_CLK will have a period 'DIV + 1' times that of the auxiliary clock 8. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the LFAST_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.42 Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0: CLK_OUT1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 928h offset = 928h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0								DIV						
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC9_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 9 divider 0 1 Enable auxiliary clock 9 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	**Divider Division Value** — The resultant CLK_OUT1 will have a period 'DIV + 1' times that of the auxiliary clock 9. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the CLK_OUT1 remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.43 Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)

This register is used to select the current clock source for the following clock.

- divided by auxiliary clock 10 divider 0: ENET_CLK

Address: 0h base + 940h offset = 940h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC10_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 10 Source Selection Control** — This value selects the current source for auxiliary clock 10. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 RMII_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.44 Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Address: 0h base + 944h offset = 944h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC10_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	**Auxiliary Clock 10 Source Selection Status** — This value indicates the current source for auxiliary clock 10. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 RMII_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.45 Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0: ENET_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 948h offset = 948h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC10_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider Division Value** — The resultant ENET_CLK will have a period 'DIV + 1' times that of the auxiliary clock 10. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the ENET_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.46 Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

This register is used to select the current clock source for the following clocks.

- undivided: (unused)
- divided by auxiliary clock 11 divider 0: ENET_TIME_CLK

Address: 0h base + 960h offset = 960h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

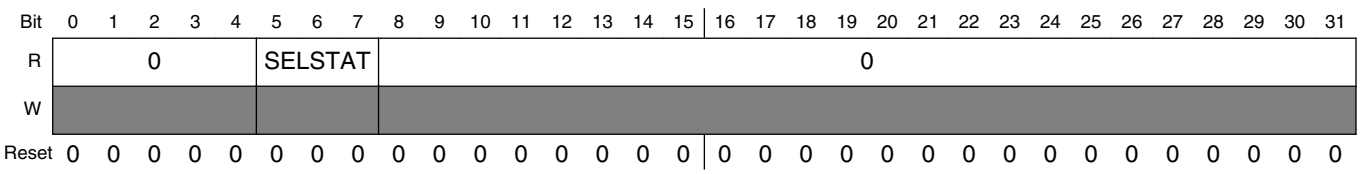
CGM_AC11_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 11 Source Selection Control** — This value selects the current source for auxiliary clock 11. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 RMII_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.47 Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

This register provides the current auxiliary clock 11 source selection.

Address: 0h base + 964h offset = 964h



CGM_AC11_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 11 Source Selection Status** — This value indicates the current source for auxiliary clock 11. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 RMII_CLK 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

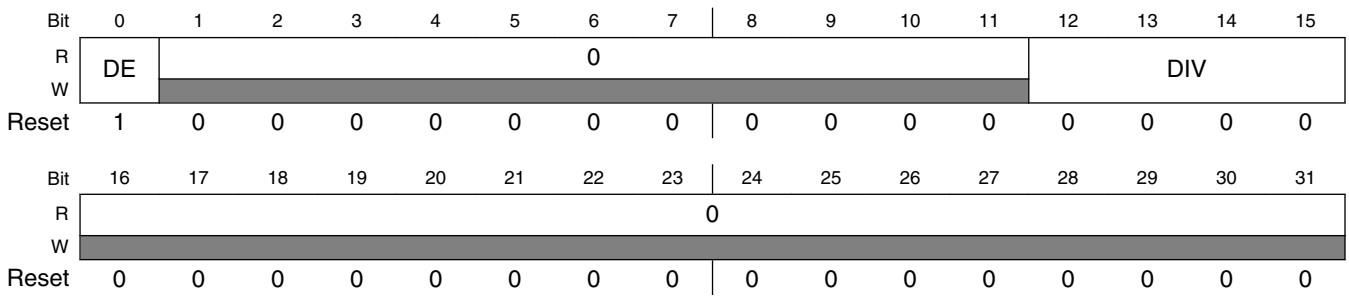
34.3.48 Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

This register controls auxiliary clock 11 divider 0: ENET_TIME_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 968h offset = 968h



CGM_AC11_DC0 field descriptions

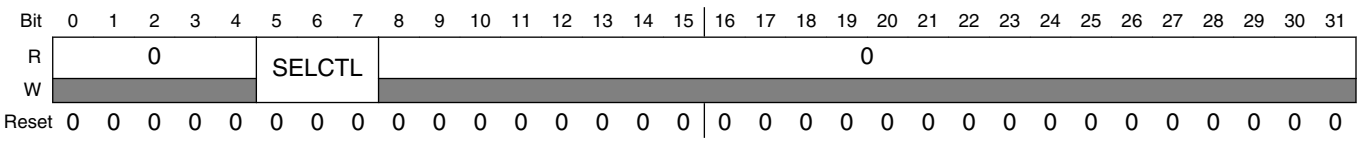
Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 11 divider 0 1 Enable auxiliary clock 11 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	**Divider Division Value** — The resultant ENET_TIME_CLK will have a period 'DIV + 1' times that of the auxiliary clock 11. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the ENET_TIME_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.49 Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)

This register is used to select the current clock source for the following clocks.

- undivided: (unused)
- divided by auxiliary clock 12 divider 0: SPI_CLK

Address: 0h base + 980h offset = 980h



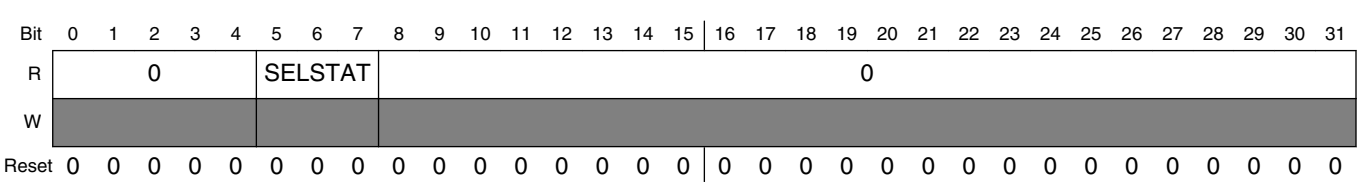
CGM_AC12_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 12 Source Selection Control** — This value selects the current source for auxiliary clock 12. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 Reserved 110 Reserved 111 Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.50 Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)

This register provides the current auxiliary clock 12 source selection.

Address: 0h base + 984h offset = 984h



CGM_AC12_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 12 Source Selection Status** — This value indicates the current source for auxiliary clock 12. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 MC_CLK 110 PBRIDGE_0_CLK 111 Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.51 Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)

This register controls auxiliary clock 12 divider 0: SPI_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 988h offset = 988h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC12_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 12 divider 0 1 Enable auxiliary clock 12 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	**Divider Division Value** — The resultant SPI_CLK will have a period 'DIV + 1' times that of the auxiliary clock 12. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the SPI_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.52 Auxiliary Clock 13 Select Control Register (CGM_AC13_SC)

This register is used to select the current clock source for the following clocks.

- undivided: (unused)
- divided by auxiliary clock 13 divider 0: LIN_CLK

Address: 0h base + 9A0h offset = 9A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC13_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	**Auxiliary Clock 13 Source Selection Control** — This value selects the current source for auxiliary clock 13. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 MC_CLK 110 PBRIDGE_0_CLK 111 Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.3.53 Auxiliary Clock 13 Select Status Register (CGM_AC13_SS)

This register provides the current auxiliary clock 13 source selection.

Address: 0h base + 9A4h offset = 9A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CGM_AC13_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	**Auxiliary Clock 13 Source Selection Status** — This value indicates the current source for auxiliary clock 13. 000 IRC_CLK 001 XOSC_CLK 010 PLL0_PHI_CLK 011 SDPLL_CLK 100 PLL1_PHI_CLK 101 MC_CLK 110 PBRIDGE_0_CLK 111 Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

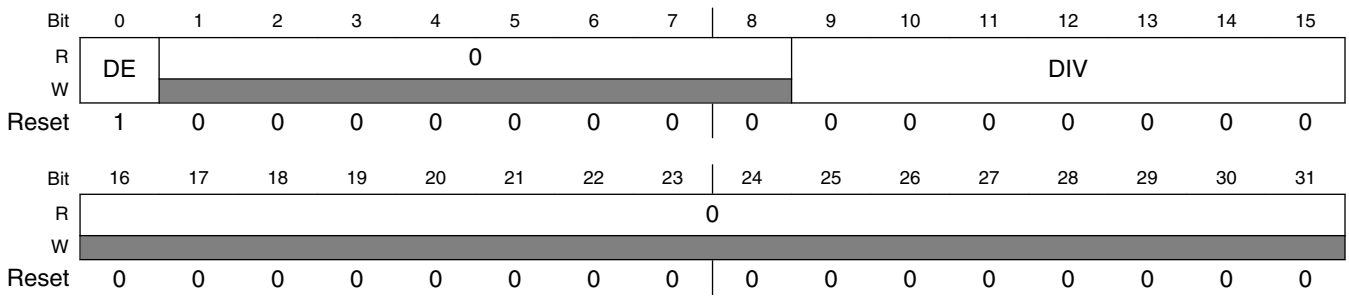
34.3.54 Auxiliary Clock 13 Divider 0 Configuration Register (CGM_AC13_DC0)

This register controls auxiliary clock 13 divider 0: LIN_CLK.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 9A8h offset = 9A8h



CGM_AC13_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 13 divider 0 1 Enable auxiliary clock 13 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	**Divider Division Value** — The resultant LIN_CLK will have a period 'DIV + 1' times that of the auxiliary clock 13. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the LIN_CLK remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

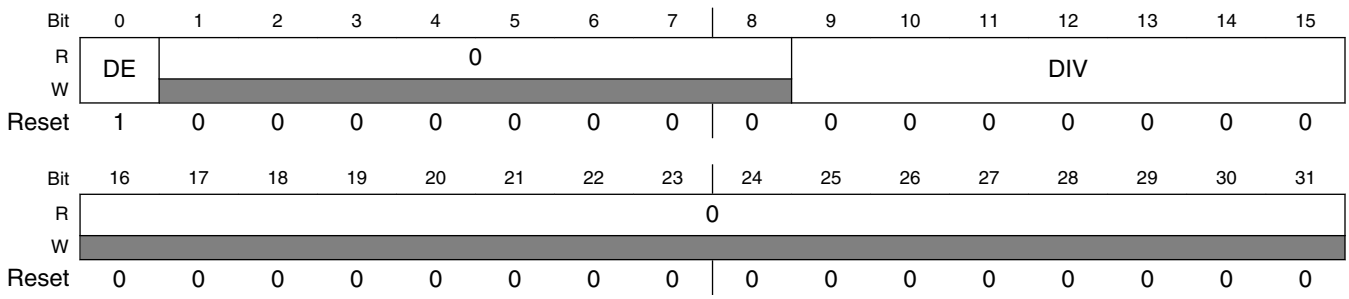
34.3.55 Auxiliary Clock 14 Divider 0 Configuration Register (CGM_AC14_DC0)

This register controls auxiliary clock 14 divider 0: CLK_OUT0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 9C8h offset = 9C8h



CGM_AC14_DC0 field descriptions

Field	Description
0 DE	**Divider Enable** 0 Disable auxiliary clock 14 divider 0 1 Enable auxiliary clock 14 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	**Divider Division Value** — The resultant CLK_OUT0 will have a period 'DIV + 1' times that of the auxiliary clock 14. If DE is set to '0' (divider 0 is disabled), any write access to the DIV field is ignored and the CLK_OUT0 remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.4 Functional Description

34.4.1 System Clock Generation

The following figure shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the IRC_CLK as the system clock and to ignore the system clock select.

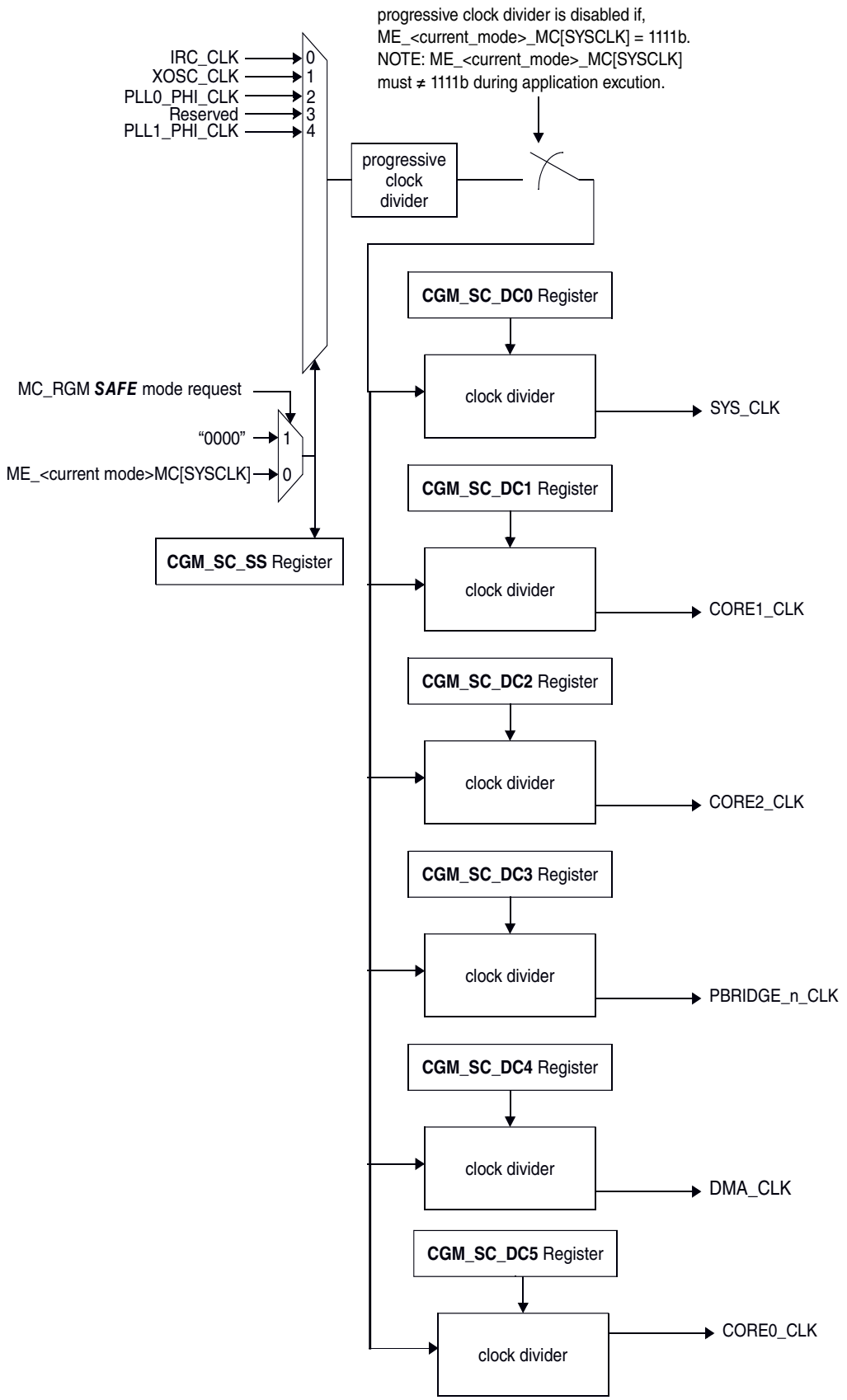


Figure 34-2. MC_CGM System Clock Generation Overview

34.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled

- on a ***SAFE*** mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

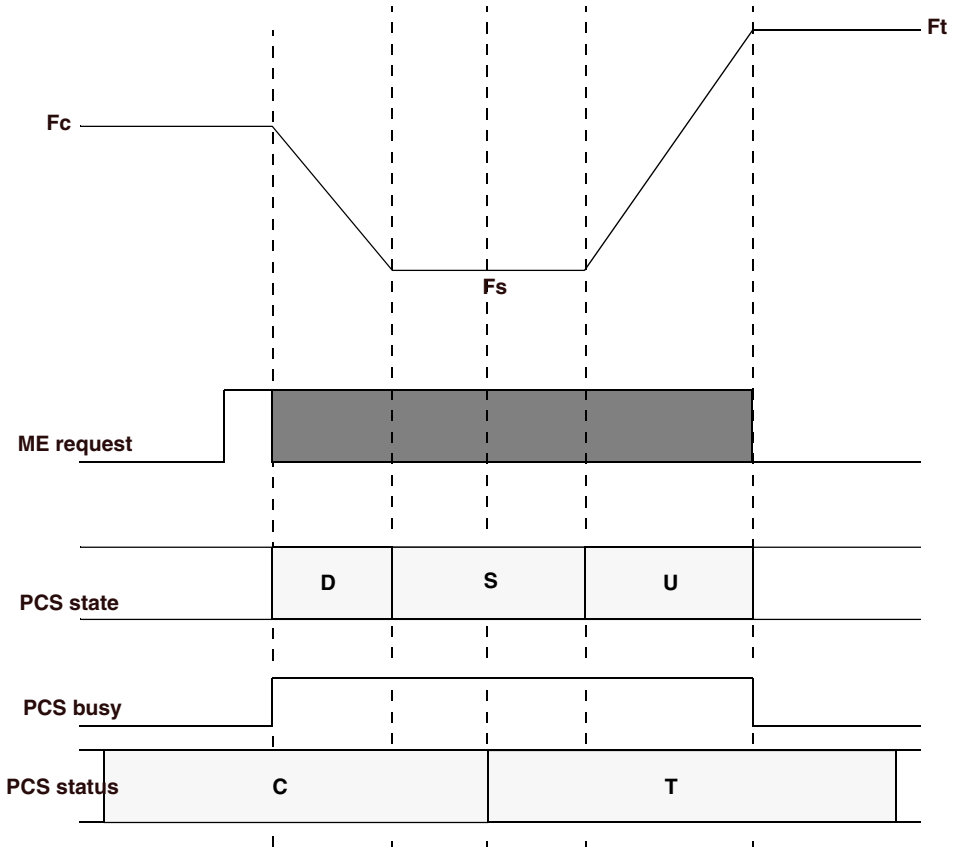
34.4.1.2 Progressive System Clock Switching

Progressive clock switching (PCS) is an option which simplifies the load changes when transitioning between modes by first going to a lower frequency, of the 16 MHz IRC, and then ramping the frequency with the new clock source selected.

The diagram below shows the frequency change in a device using PCS when the system moves from Fc to Ft. The current clock or Fc is gradually divided using the PCS configuration registers to reach 16 MHz which is the safe clock freq (Fs). Once the current clock frequency comes down to Fs, a clock switch to IRC is made. At this point all the peripherals and cores are switched on/off to match the target configuration of the MC_ME registers. Then the clock is switched from IRC to the target clock with dividers configured such that the target clock is running at 16 MHz (Fs). After that the dividers are changed to get a gradual ramp up to the target frequency(Ft), at this point, the PCS divider is dividing the target clock by 1.

NOTE

PCS configuration should not be done if some communication peripherals e.g., FlexCAN, LinFlex which are required to be working in a seamless manner during mode transition.



Note:
C: Current clock, T: Target clock, D: Clock Ramp down, S: Safe clock, U: Clock Ramp up

Figure 34-3. Progressive System Clock Switching

Based on the power level values of the current and target modes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up. During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVEn registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVSn registers, where n corresponds to the target system clock source selection.

34.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in the following equation.

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

Figure 34-4. Equation 1

34.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field **CGM_PCS_SDUR[SDUR]** defines the duration of one system clock switch step in terms of IRC_CLK. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

34.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the **PCS_DIVC*n*[RATE]**register is given in the following figure.

Table 34-1. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d = CGM_PCS_DIVC*n*[RATE]
0.05	12
0.10	48
0.15	112
0.20	184

34.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in the following equation.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d/1000}} \right\rceil$$

Figure 34-5. Equation 2

34.4.1.2.5 Clock Ramp-Down

The clock ramp-down starts with the divider value 1 and with the given divider increment value **PCD_DIVC*n*[RATE]** and ends with the given divider value **PCS_DIVE*n*[DIVE]**.

The divider end value for clock ramp-down is given in the following equation.

$$\text{PCS\_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Figure 34-6. Equation 3

Where f_{curr} is the frequency of the currently selected system clock source.

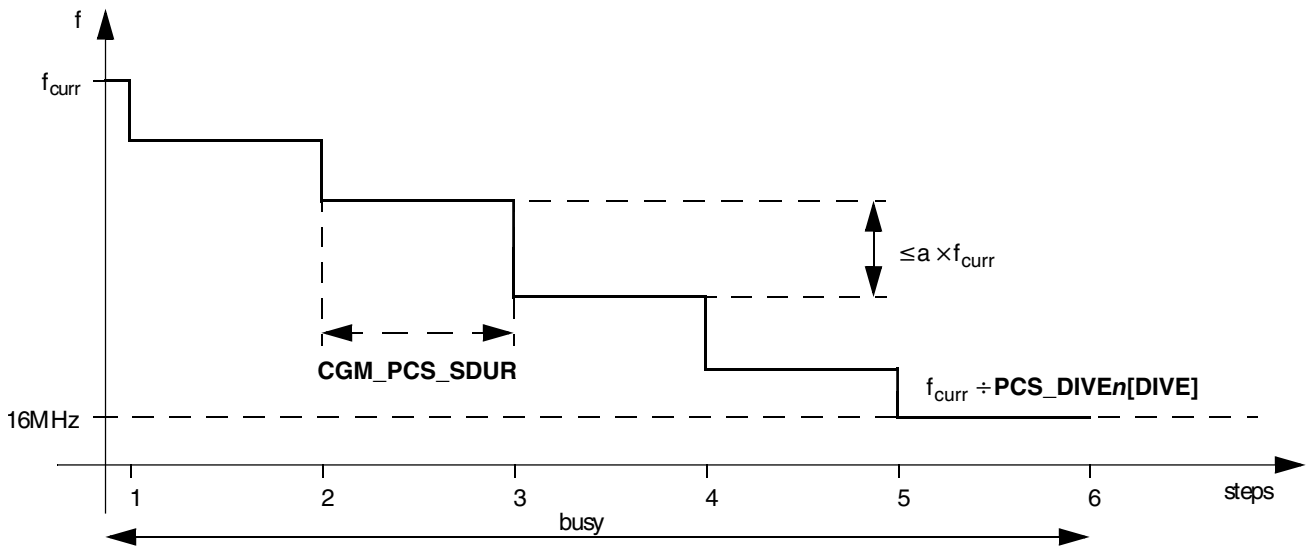


Figure 34-7. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

34.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value **PCS_DIVS*n*[DIVS]** and with the given divider decrement value **PCD_DIVC*n*[INIT]** and ends with the divider value 1.

The initial divider change start value **PCS_DIVC*n*[INIT]** for system clock ramp-up is given in Figure 34-8.

$$\text{PCS\_DIVCn[INIT]} = d \times k$$

Figure 34-8. Equation 4

Where *k* is calculated from Figure 34-5 using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in Figure 34-9.

$$\text{PCS\_DIVSn[DIVS]} = d\,\frac{k\,(k+1)}{2} + 999$$

Figure 34-9. Equation 5

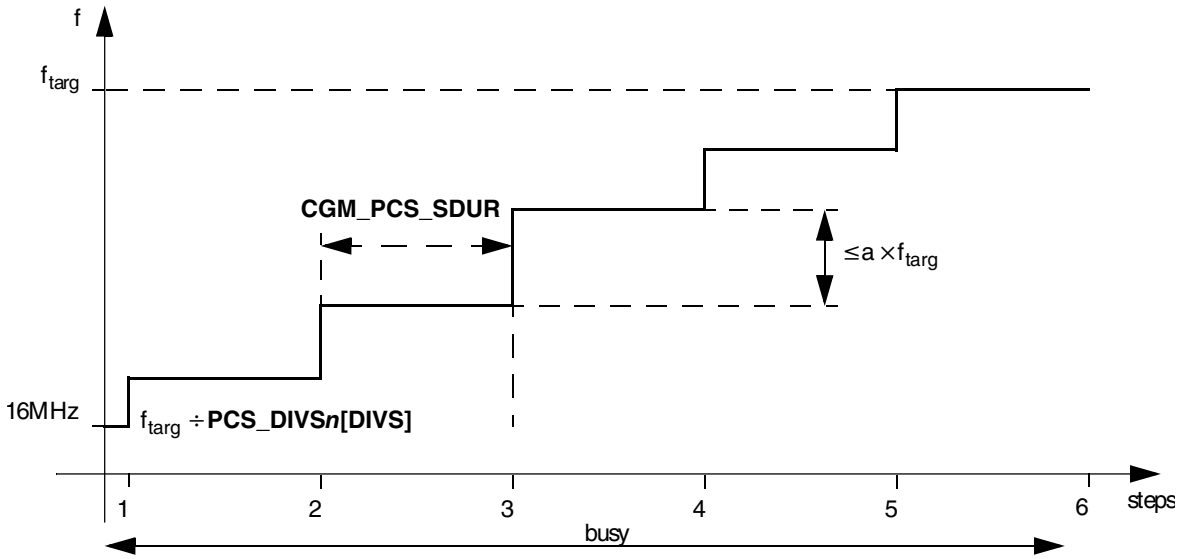


Figure 34-10. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

34.4.1.3 System Clock Disable

During the TEST mode, the system clock can be disabled by the MC_ME.

34.4.1.4 System Clock Divider Synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

Warning

Configuring the system clock dividers incorrectly will cause the divided clocks to become unsynchronized. This may lead to lost communication between the clock domains and/or the inability to further change the system clock divider division factors without first disabling them or performing a reset.

34.4.1.4.1 Example 1: Correct System Clock Divider Configuration

Table 34-2. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** =0

Table continues on the next page...

Table 34-2. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization (continued)

Divider	Division Factor	Register Value
1	2	**CGM_SC_DC1[DIV]** = 1
2	6	**CGM_SC_DC2[DIV]** = 5

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 2 is an integer multiple of 1, its configration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

34.4.1.4.2 Example 2: Incorrect System Clock Divider Configuration

Table 34-3. MC_CGM Example System Clock Division Values Incompatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 0
1	4	**CGM_SC_DC1[DIV]** = 3
2	6	**CGM_SC_DC2[DIV]** = 5

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 4 is an integer multiple of 1, its configration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is not an integer multiple of 4, its configuration is incorrect.

34.4.2 Auxiliary Clock Generation

The following figures show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)
- Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)
- Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)
- Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)
- Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)
- Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)
- Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)
- Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)
- Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

- Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)
- Auxiliary Clock 13 Select Control Register (CGM_AC13_SC)

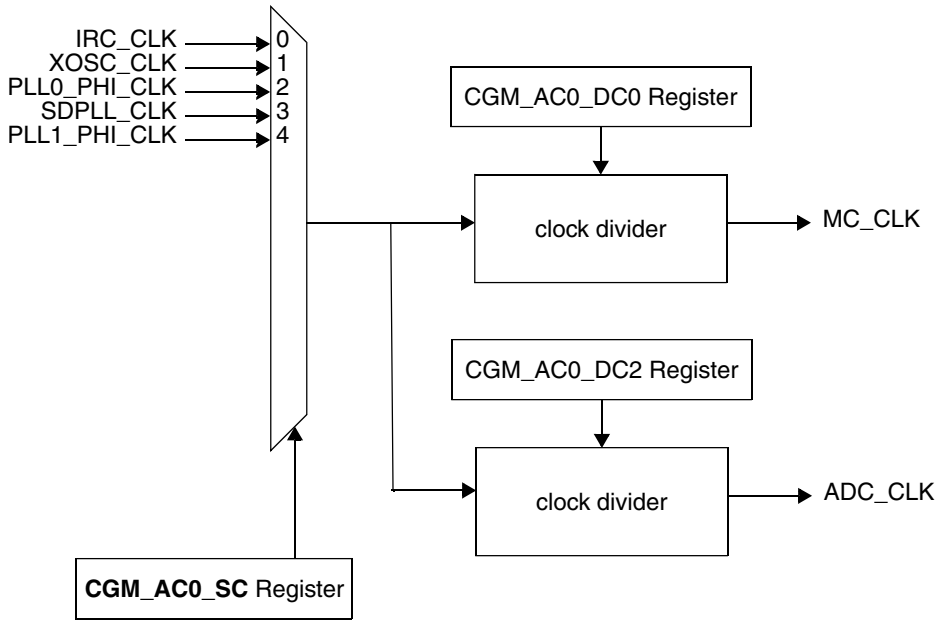


Figure 34-11. MC_CGM Auxiliary Clock 0 Generation Overview

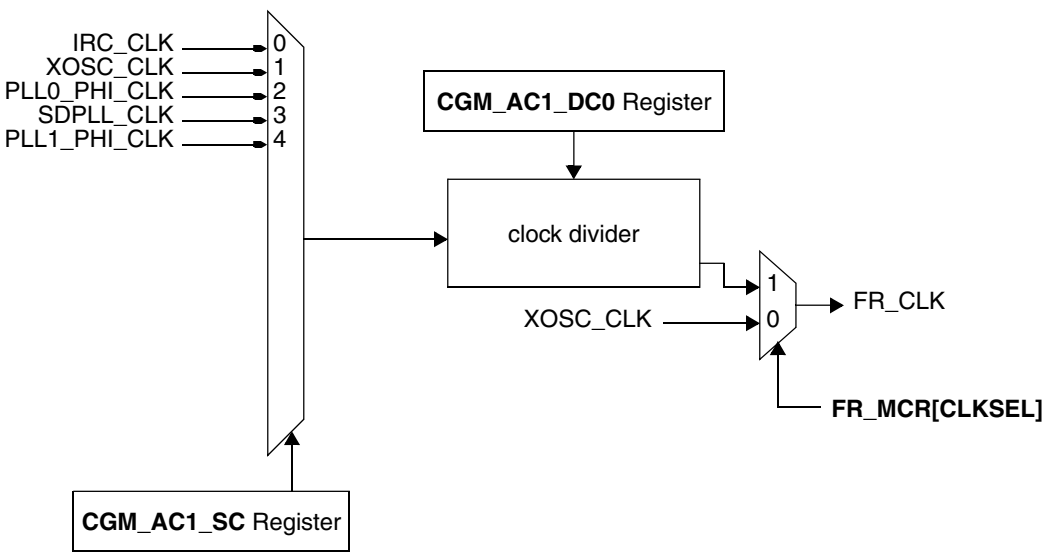


Figure 34-12. MC_CGM Auxiliary Clock 1 Generation Overview

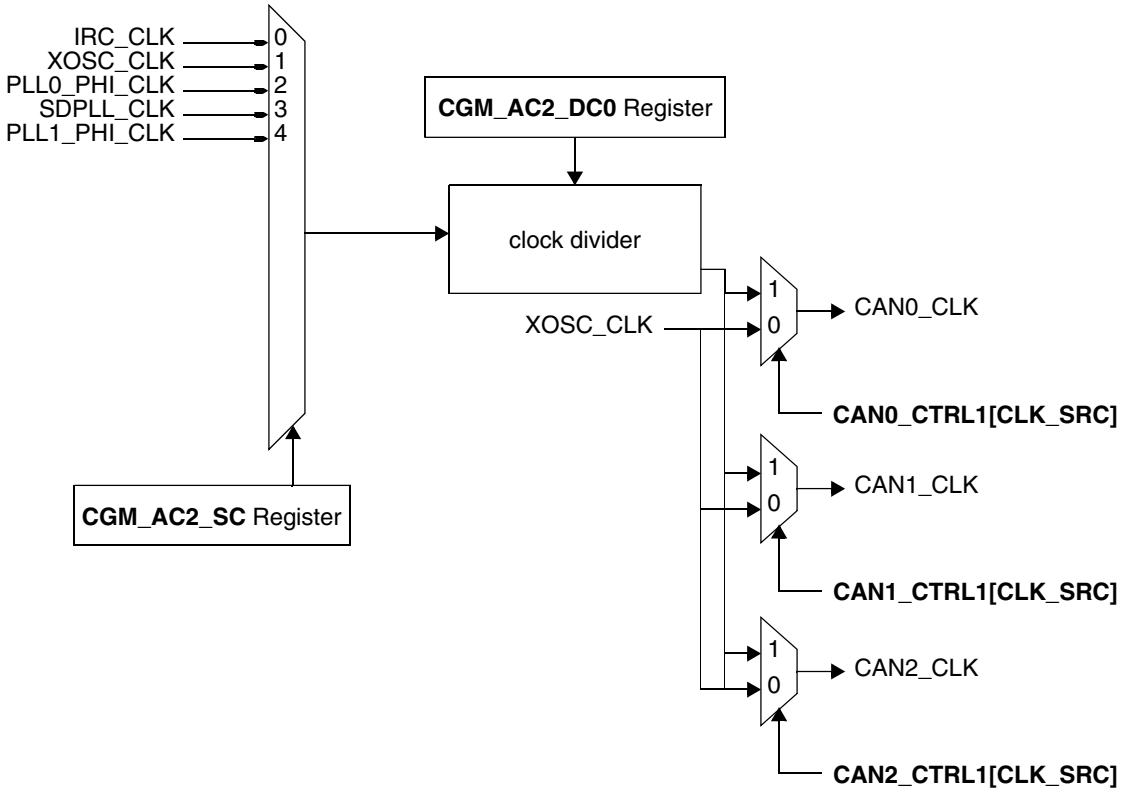


Figure 34-13. MC_CGM Auxiliary Clock 2 Generation Overview

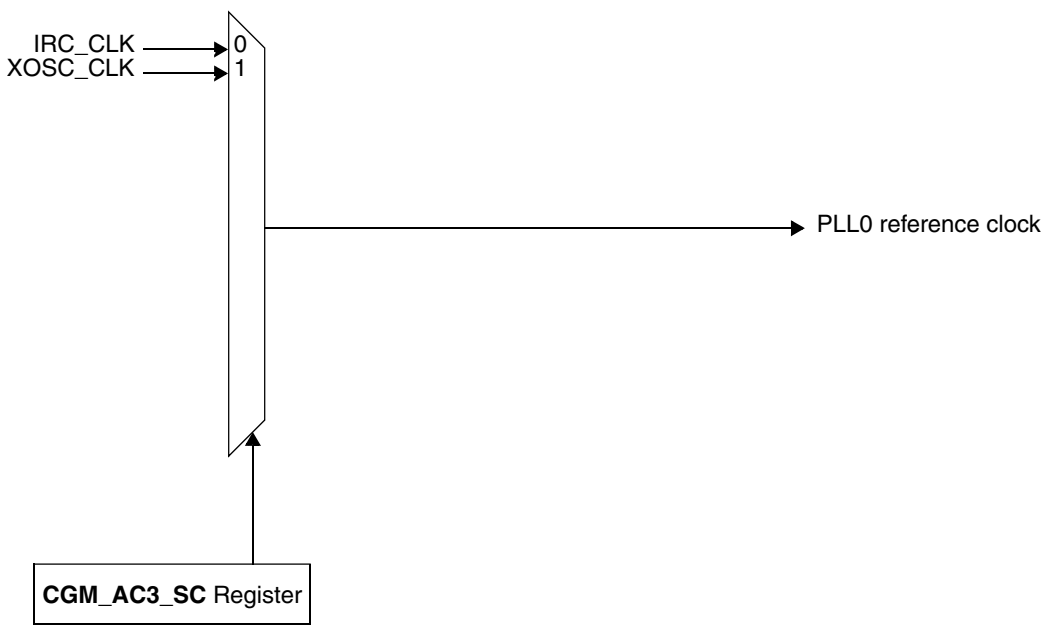


Figure 34-14. MC_CGM Auxiliary Clock 3 Generation Overview

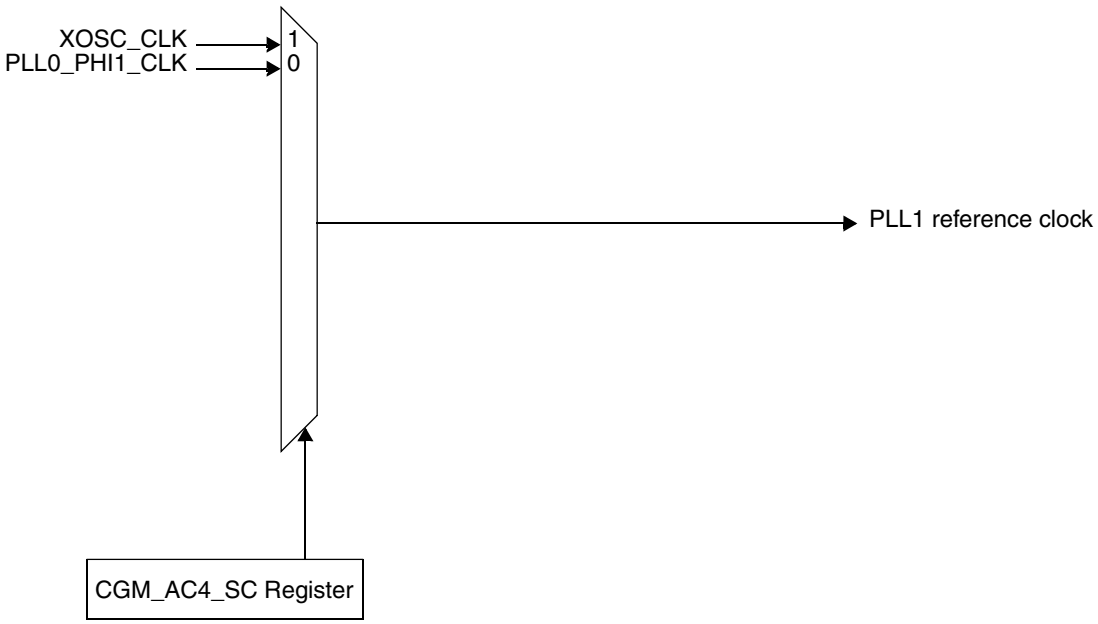


Figure 34-15. MC_CGM Auxiliary Clock 4 Generation Overview

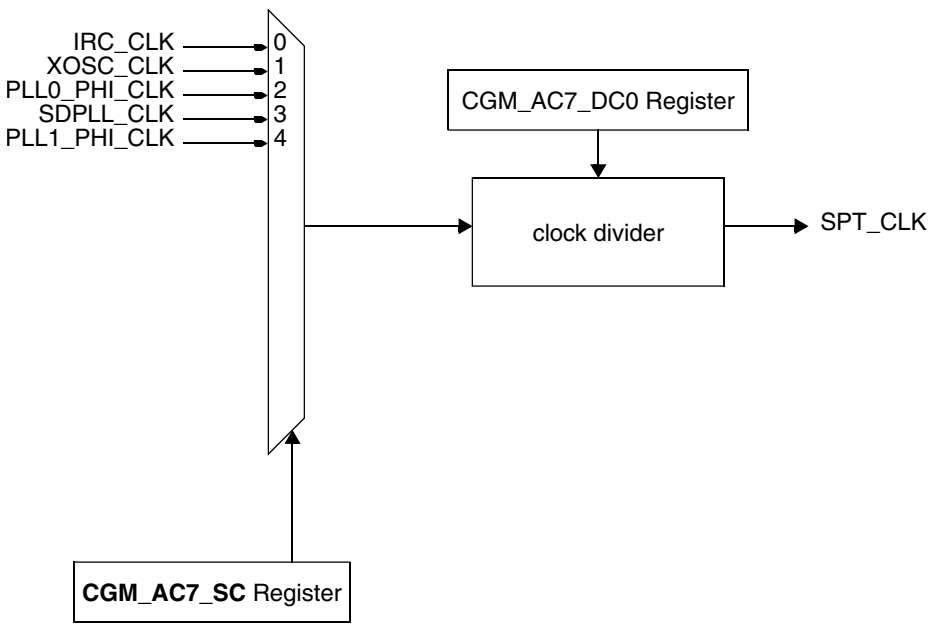


Figure 34-16. MC_CGM Auxiliary Clock 7 Generation Overview

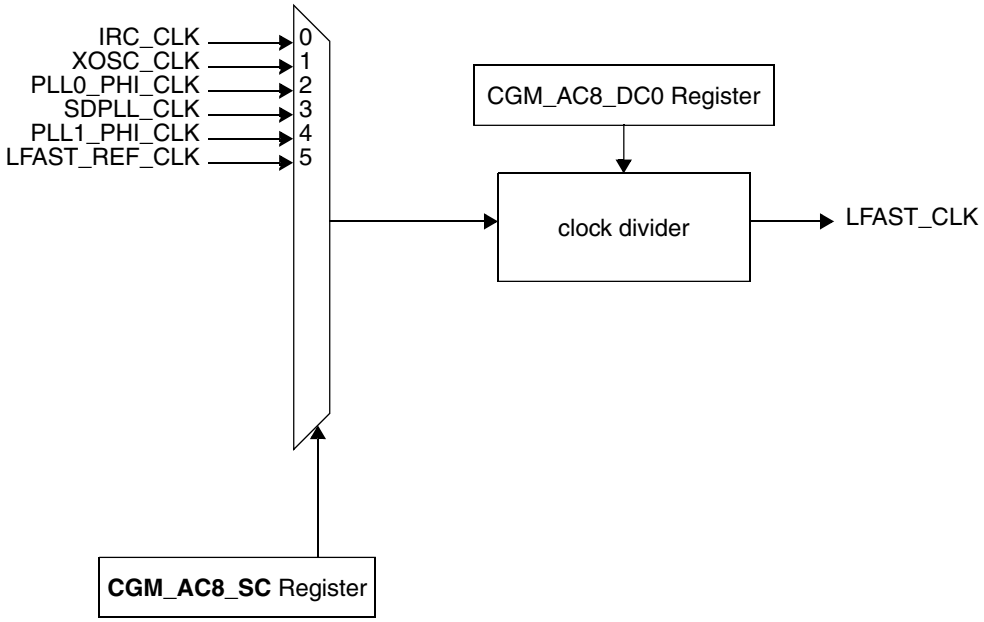


Figure 34-17. MC_CGM Auxiliary Clock 8 Generation Overview

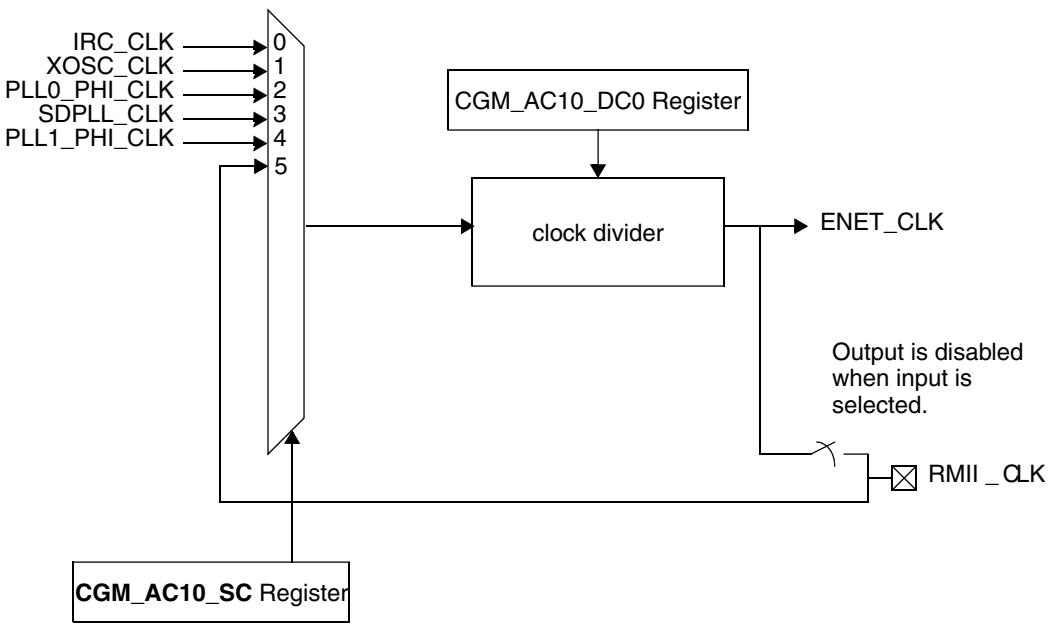


Figure 34-18. MC_CGM Auxiliary Clock 10 Generation Overview

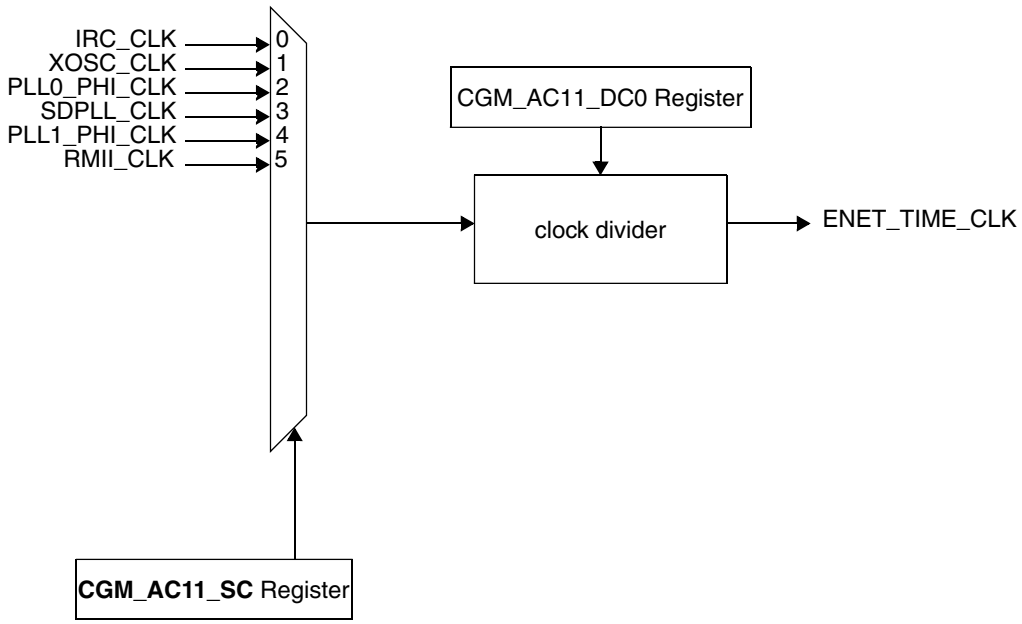


Figure 34-19. MC_CGM Auxiliary Clock 11 Generation Overview

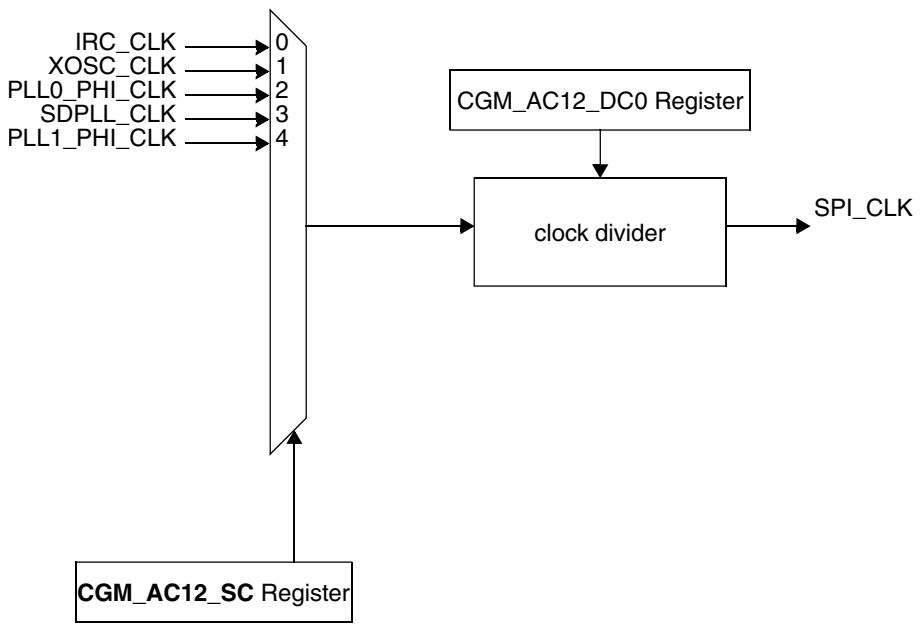


Figure 34-20. MC_CGM Auxiliary Clock 12 Generation Overview

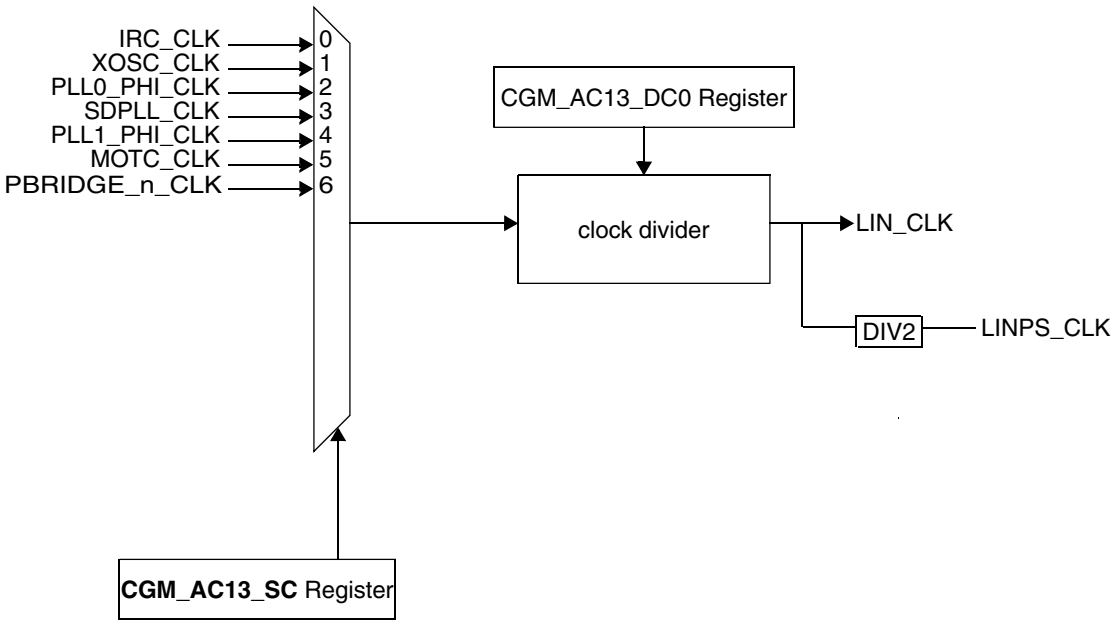


Figure 34-21. MC_CGM Auxiliary Clock 13 Generation Overview

NOTE

The selection mux is not glitch-less. Therefore, the auxiliary clock must be disabled in the MC_CGM before the time of selection until a 'safe' time (few clock cycles) afterwards in order to guarantee that no glitches are propagated.

34.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System Clock Divider 0 Configuration Register (CGM_SC_DC0)
- System Clock Divider 1 Configuration Register (CGM_SC_DC1)
- System Clock Divider 2 Configuration Register (CGM_SC_DC2)
- System Clock Divider 3 Configuration Register (CGM_SC_DC3)
- System Clock Divider 4 Configuration Register (CGM_SC_DC4)
- System Clock Divider 5 Configuration Register (CGM_SC_DC5)
- Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)
- Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)
- Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

- Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)
- Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)
- Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)
- Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)
- Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)
- Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)
- Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)
- Auxiliary Clock 13 Divider 0 Configuration Register (CGM_AC13_DC0)
- Auxiliary Clock 14 Divider 0 Configuration Register (CGM_AC14_DC0)

The reset value of all counters is ‘1’. If a divider has its DE bit in the respective configuration register set to ‘0’ (the divider is disabled), any value in its DIV field is ignored.

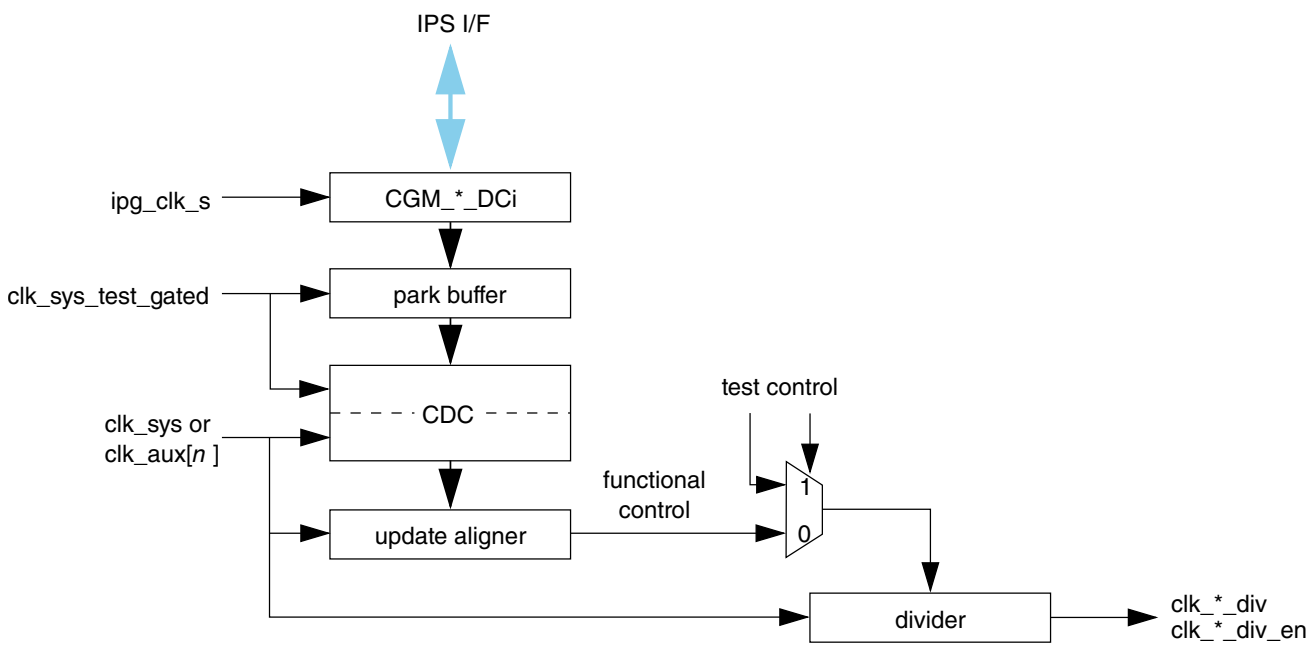


Figure 34-22. MC_CGM General Divider Structure Overview

34.4.3.1 Fractional Divider Details

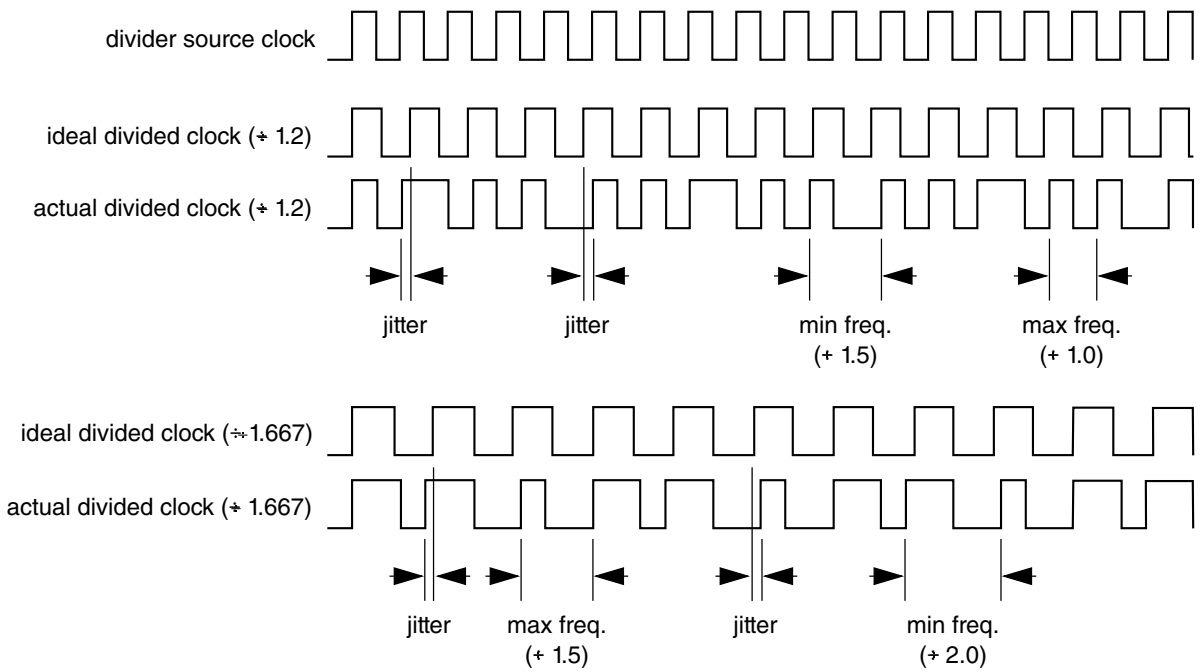


Figure 34-23. MC_CGM Fractional Clock Division Examples

For division factors of *x*.0, where $x > 0$, the fractional divider does not introduce any additional jitter to that of the clock source, neither on the rising edge nor on the falling edge, and the duty cycle = 50%.

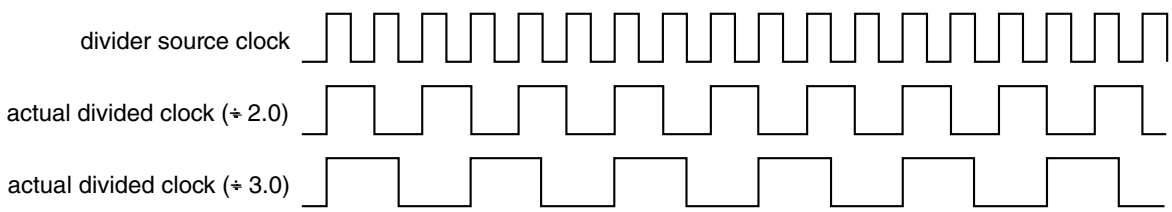


Figure 34-24. MC_CGM Fractional Clock Division Examples (x.0 division factor case)

For division factors of *x*.5, where $x > 0$, the fractional divider does not introduce any additional jitter to that of the clock source, neither on the rising edge nor on the falling edge, and the duty cycle is a constant value > 50%.

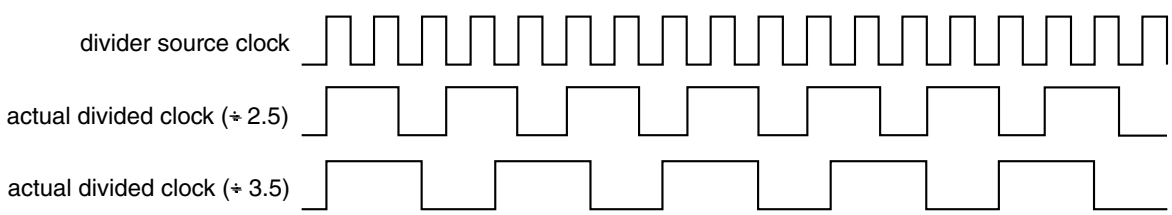


Figure 34-25. MC_CGM Fractional Clock Division Examples (x.5 division factor case)

34.4.3.2 Divider Update Synchronization

For the system clock all the associated dividers of the punch-out and 50% duty cycle types are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

Table 34-4 shows an example of a correct divider configuration.

Table 34-4. MC_CGM Example Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	CGM_xC_DC0[DIV] = 0
1	2	CGM_xC_DC1[DIV] = 1
2	6	CGM_xC_DC2[DIV] = 5

In this case, clock divider 1 has a division factor greater than that of clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct. Also, clock divider 3 has a division factor that is greater than those of both clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

Table 34-5 shows an example of an incorrect divider configuration.

Table 34-5. MC_CGM Example Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	CGM_xC_DC0[DIV] = 0
1	4	CGM_xC_DC1[DIV] = 3
2	6	CGM_xC_DC2[DIV] = 5

In this case, clock divider 1 has a division factor greater than that of clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct. Also, clock divider 3 has a division factor that is greater than those of both clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

WARNING

Configuring the clock dividers incorrectly will cause the divided clocks to become unsynchronized. This may lead to lost communication between the clock domains and/or the inability to further change the clock divider division factors without first disabling them or performing a reset.

34.4.3.3 Divider Update Triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the UPD_STAT register is set. this bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

34.4.3.3.1 Immediate Divider Update

If a divider group has been configured to be updated immediately (i.e., the corresponding DIV_UPD_TYPE register bit = ‘0’), the configuration of a divider in that group will be applied immediately on the write to that divider's configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

34.4.3.3.2 Software-Triggered Divider Update

If a divider group has been configured to be updated via a software trigger (i.e., the corresponding DIV_UPD_TYPE register bit = ‘1’), the configuration of a divider in that group will not be applied immediately on the write to that divider's configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

The following software programming steps may be used for updating the system clock dividers:

1. Configure the SC_DC***n*** registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in Divider Update Synchronization.
2. Set the DIV_UPD_TYPE.SYS_UPD_TYPE field as ’1’ to enable the update via a software trigger
3. Optional Step: If the divider ratios are not changing with respect to the previous configuration, the SC_DIV_RC.SYS_DIV_RATIO_CHNG field may be reset to 0 to disable the crossbar halt handshake mechanism

4. Write any non-zero value to the DIV_UPD_TRIG register to trigger the divider update.
5. Poll the status register DIV_UPD_STAT.SYS_UPD_STAT field to check when the divider update completes.

NOTE

In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (i.e., all relevant bits in the DIV_UPD_STAT register are ‘0’) before triggering a new update via the DIV_UPD_TRIG register

Chapter 35
IRCOSC Digital Interface

35.1 Introduction

The Internal RC Oscillator Digital Interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

35.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see the Data Sheet for timing details) which is required for stabilization. The IRCOSC status is updated in the MC_ME_GS[S_IRC] field (see the "Mode Entry Module MC_ME" chapter).

The IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

35.3 Memory map and register definition

IRCOSC memory map

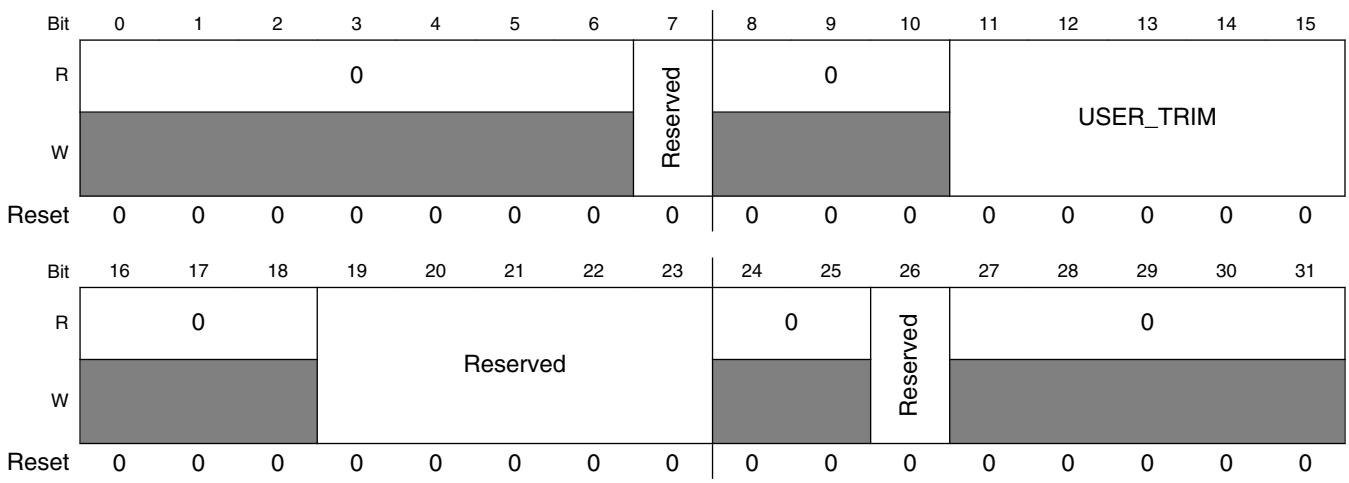
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	See section	35.3.1/1232

35.3.1 IRCOSC Control Register (IRCOSC_CTL)

The IRCOSC_CTL contains user programmable parameters.

IRCOSC_CTL is writable only in supervisor mode.

Address: 0h base + 0h offset = 0h



IRCOSC_CTL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This bit can be written with any value, but writes are ignored. A read returns last written value.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. Frequency trimming calculation shows how RCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IRCOSC_CTL field descriptions (continued)

Field	Description
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This field may sometimes have the value of 1, and can be cleared by writing a 1 to this bit. However, this field has no effect on the IRCOSC or chip behavior.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.3.2 Frequency trimming calculation

Table 35-1. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	....
10011	$f + (3 \times \delta f_{TRIM})$
10010	$f + (2 \times \delta f_{TRIM})$
10001	$f + \delta f_{TRIM}$
10000	f
00000	f
00001	$f - \delta f_{TRIM}$
00010	$f - (2 \times \delta f_{TRIM})$
00011	$f - (3 \times \delta f_{TRIM})$
....	....

NOTE

See the data sheet for δf_{TRIM} value and min/max frequency variations.

Chapter 36
Enhanced Direct Memory Access (eDMA)

36.1 DMA Controller configuration

Due to the strict architecture of the RAM controller, the DMA_DCHMIDn register should not be used to replicate Core_1 or Core_2 accesses to the SRAM from the DMA. Doing so results in an error that is reflected in the DMA_ES register.

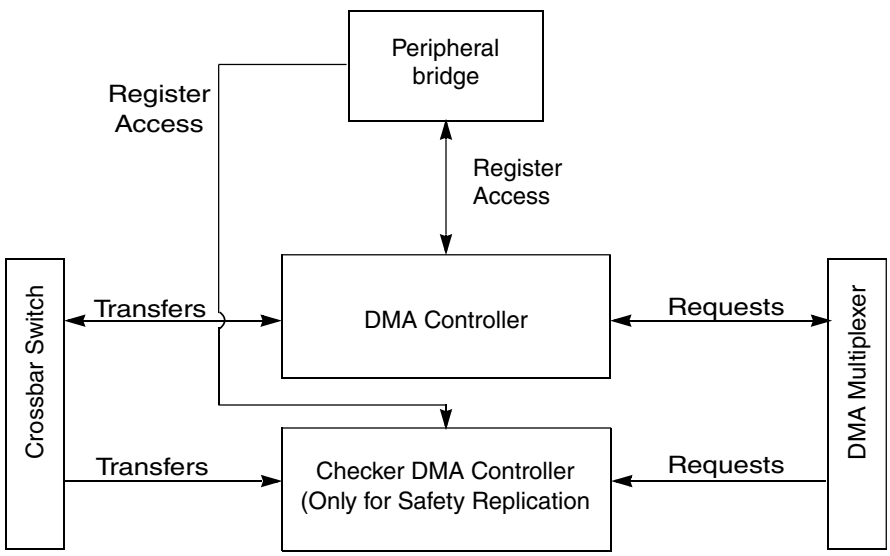


Figure 36-1. DMA controller configuration

36.2 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:

- Source address and destination address calculations
- Data-movement operations

• Local memory containing transfer control descriptors for each of the 32 channels

36.2.1 eDMA system block diagram

Figure 36-2 illustrates the components of the eDMA system, including the eDMA module ("engine").

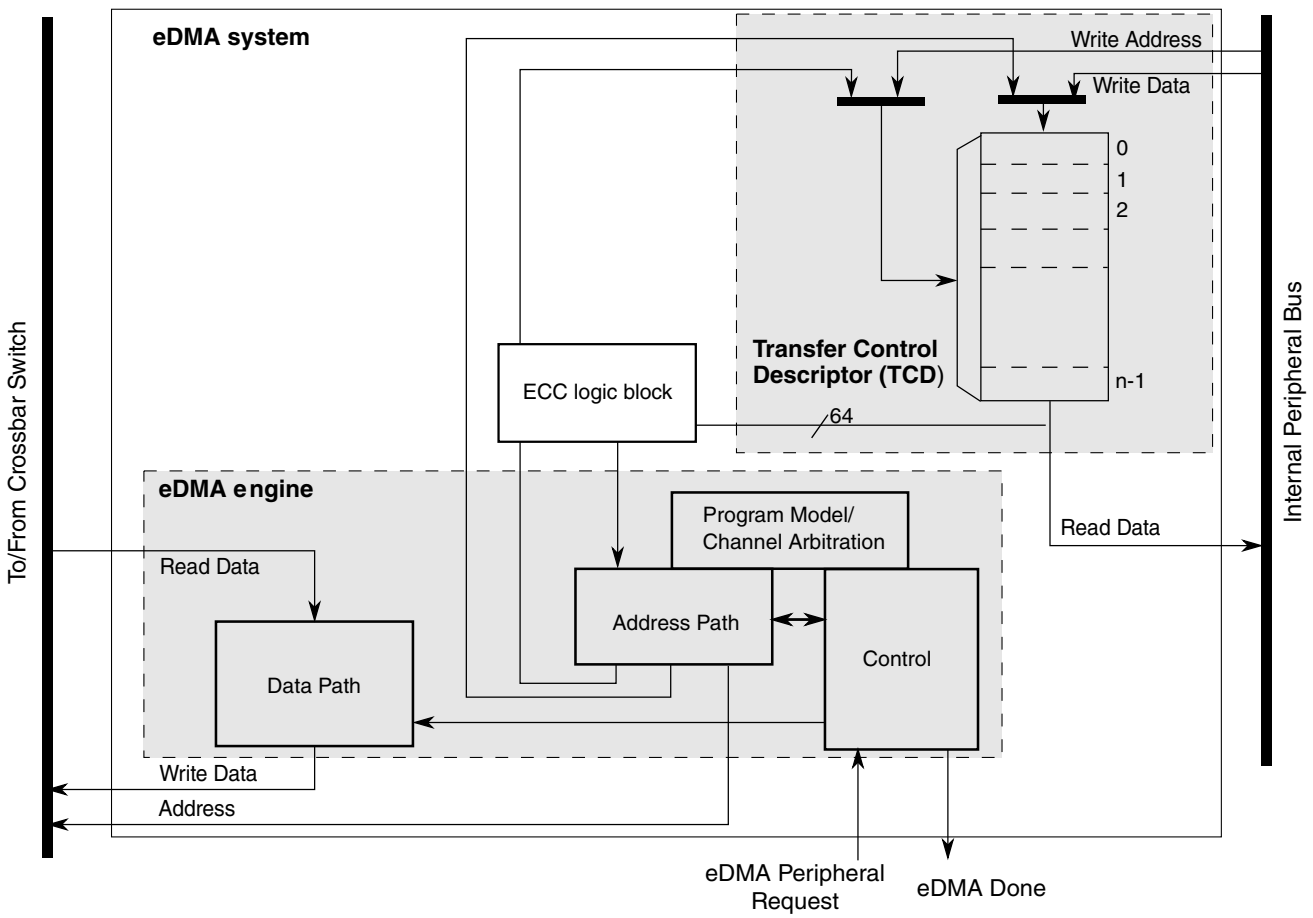


Figure 36-2. eDMA system block diagram

36.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 36-1. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 36-2. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

36.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing

- Support for complex data structures
- Error detection and error correction

In the discussion of this module, *n* is used to reference the channel number.

36.3 Modes of operation

The eDMA operates in the following modes:

Table 36-3. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

36.4 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

36.4.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

36.4.2 TCD initialization

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

36.4.3 TCD structure

DMA Basics: TCD Structure
- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD

Word Offset	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
0x1000	SADDR							
0x1004	SMOD	SSIZE	DMOD	DSIZE	SOFF			
0x1008	NBYTES[1]							
0x1008	SMLOE[1] / DMLOE[1]	MLOFF or NBYTES[1]					NBYTES[1]	
0x100C	SLAST							
0x1010	DADDR							
0x1014	CITER.E_LINK	CITER or CITER.LINKCH	CITER		DOFF			
0x1018	DLAST_SGA							
0x101C	BITER.E_LINK	BITER or BITER.LINKCH	BITER		BWC	MAJOR LINKCH	DONE / ACTIVE / MAJOR.E_LINK / E_SG / D_REQ / INT_HALF / INT_MAJ / START	

[1] The fields implemented in Word 2 depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

36.4.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

NOTE

Accessing locations at offsets 38h and 3Ch will not result in a transfer error. Location 38h behaves as ROZ. Location 3Ch behaves as R/W.

DMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (DMA_CR)	32	R/W	0000_0400h	36.4.5/1268
4	Error Status Register (DMA_ES)	32	R	0000_0000h	36.4.6/1271
C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	36.4.7/1273
14	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	36.4.8/1277
18	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	36.4.9/1280
19	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	36.4.10/ 1281
1A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	36.4.11/ 1282
1B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	36.4.12/ 1283
1C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	36.4.13/ 1284
1D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	36.4.14/ 1285
1E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	36.4.15/ 1286
1F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	36.4.16/ 1287
24	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	36.4.17/ 1288
2C	Error Register (DMA_ERR)	32	R/W	0000_0000h	36.4.18/ 1291

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	36.4.19/ 1295
100	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	36.4.20/ 1301
101	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	36.4.20/ 1301
102	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	36.4.20/ 1301
103	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	36.4.20/ 1301
104	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	36.4.20/ 1301
105	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	36.4.20/ 1301
106	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	36.4.20/ 1301
107	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	36.4.20/ 1301
108	Channel n Priority Register (DMA_DCHPRI8)	8	R/W	See section	36.4.20/ 1301
109	Channel n Priority Register (DMA_DCHPRI9)	8	R/W	See section	36.4.20/ 1301
10A	Channel n Priority Register (DMA_DCHPRI10)	8	R/W	See section	36.4.20/ 1301
10B	Channel n Priority Register (DMA_DCHPRI11)	8	R/W	See section	36.4.20/ 1301
10C	Channel n Priority Register (DMA_DCHPRI12)	8	R/W	See section	36.4.20/ 1301
10D	Channel n Priority Register (DMA_DCHPRI13)	8	R/W	See section	36.4.20/ 1301
10E	Channel n Priority Register (DMA_DCHPRI14)	8	R/W	See section	36.4.20/ 1301
10F	Channel n Priority Register (DMA_DCHPRI15)	8	R/W	See section	36.4.20/ 1301
110	Channel n Priority Register (DMA_DCHPRI16)	8	R/W	See section	36.4.20/ 1301
111	Channel n Priority Register (DMA_DCHPRI17)	8	R/W	See section	36.4.20/ 1301
112	Channel n Priority Register (DMA_DCHPRI18)	8	R/W	See section	36.4.20/ 1301
113	Channel n Priority Register (DMA_DCHPRI19)	8	R/W	See section	36.4.20/ 1301
114	Channel n Priority Register (DMA_DCHPRI20)	8	R/W	See section	36.4.20/ 1301

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
115	Channel n Priority Register (DMA_DCHPRI21)	8	R/W	See section	36.4.20/ 1301
116	Channel n Priority Register (DMA_DCHPRI22)	8	R/W	See section	36.4.20/ 1301
117	Channel n Priority Register (DMA_DCHPRI23)	8	R/W	See section	36.4.20/ 1301
118	Channel n Priority Register (DMA_DCHPRI24)	8	R/W	See section	36.4.20/ 1301
119	Channel n Priority Register (DMA_DCHPRI25)	8	R/W	See section	36.4.20/ 1301
11A	Channel n Priority Register (DMA_DCHPRI26)	8	R/W	See section	36.4.20/ 1301
11B	Channel n Priority Register (DMA_DCHPRI27)	8	R/W	See section	36.4.20/ 1301
11C	Channel n Priority Register (DMA_DCHPRI28)	8	R/W	See section	36.4.20/ 1301
11D	Channel n Priority Register (DMA_DCHPRI29)	8	R/W	See section	36.4.20/ 1301
11E	Channel n Priority Register (DMA_DCHPRI30)	8	R/W	See section	36.4.20/ 1301
11F	Channel n Priority Register (DMA_DCHPRI31)	8	R/W	See section	36.4.20/ 1301
140	Channel n Master ID Register (DMA_DCHMID0)	8	R/W	00h	36.4.21/ 1302
141	Channel n Master ID Register (DMA_DCHMID1)	8	R/W	00h	36.4.21/ 1302
142	Channel n Master ID Register (DMA_DCHMID2)	8	R/W	00h	36.4.21/ 1302
143	Channel n Master ID Register (DMA_DCHMID3)	8	R/W	00h	36.4.21/ 1302
144	Channel n Master ID Register (DMA_DCHMID4)	8	R/W	00h	36.4.21/ 1302
145	Channel n Master ID Register (DMA_DCHMID5)	8	R/W	00h	36.4.21/ 1302
146	Channel n Master ID Register (DMA_DCHMID6)	8	R/W	00h	36.4.21/ 1302
147	Channel n Master ID Register (DMA_DCHMID7)	8	R/W	00h	36.4.21/ 1302
148	Channel n Master ID Register (DMA_DCHMID8)	8	R/W	00h	36.4.21/ 1302
149	Channel n Master ID Register (DMA_DCHMID9)	8	R/W	00h	36.4.21/ 1302
14A	Channel n Master ID Register (DMA_DCHMID10)	8	R/W	00h	36.4.21/ 1302

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14B	Channel n Master ID Register (DMA_DCHMID11)	8	R/W	00h	36.4.21/ 1302
14C	Channel n Master ID Register (DMA_DCHMID12)	8	R/W	00h	36.4.21/ 1302
14D	Channel n Master ID Register (DMA_DCHMID13)	8	R/W	00h	36.4.21/ 1302
14E	Channel n Master ID Register (DMA_DCHMID14)	8	R/W	00h	36.4.21/ 1302
14F	Channel n Master ID Register (DMA_DCHMID15)	8	R/W	00h	36.4.21/ 1302
150	Channel n Master ID Register (DMA_DCHMID16)	8	R/W	00h	36.4.21/ 1302
151	Channel n Master ID Register (DMA_DCHMID17)	8	R/W	00h	36.4.21/ 1302
152	Channel n Master ID Register (DMA_DCHMID18)	8	R/W	00h	36.4.21/ 1302
153	Channel n Master ID Register (DMA_DCHMID19)	8	R/W	00h	36.4.21/ 1302
154	Channel n Master ID Register (DMA_DCHMID20)	8	R/W	00h	36.4.21/ 1302
155	Channel n Master ID Register (DMA_DCHMID21)	8	R/W	00h	36.4.21/ 1302
156	Channel n Master ID Register (DMA_DCHMID22)	8	R/W	00h	36.4.21/ 1302
157	Channel n Master ID Register (DMA_DCHMID23)	8	R/W	00h	36.4.21/ 1302
158	Channel n Master ID Register (DMA_DCHMID24)	8	R/W	00h	36.4.21/ 1302
159	Channel n Master ID Register (DMA_DCHMID25)	8	R/W	00h	36.4.21/ 1302
15A	Channel n Master ID Register (DMA_DCHMID26)	8	R/W	00h	36.4.21/ 1302
15B	Channel n Master ID Register (DMA_DCHMID27)	8	R/W	00h	36.4.21/ 1302
15C	Channel n Master ID Register (DMA_DCHMID28)	8	R/W	00h	36.4.21/ 1302
15D	Channel n Master ID Register (DMA_DCHMID29)	8	R/W	00h	36.4.21/ 1302
15E	Channel n Master ID Register (DMA_DCHMID30)	8	R/W	00h	36.4.21/ 1302
15F	Channel n Master ID Register (DMA_DCHMID31)	8	R/W	00h	36.4.21/ 1302
1000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1004	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	0000h	36.4.23/ 1303
1006	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	0000h	36.4.24/ 1304
1008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
100C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1014	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1014	DMA_TCD0_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1016	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	0000h	36.4.32/ 1311
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
101E	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	0000h	36.4.36/ 1314
1020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1024	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	0000h	36.4.23/ 1303
1026	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	0000h	36.4.24/ 1304
1028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
102C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1034	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1034	DMA_TCD1_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1036	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	0000h	36.4.32/ 1311
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
103E	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	0000h	36.4.36/ 1314
1040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1044	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	0000h	36.4.23/ 1303
1046	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	0000h	36.4.24/ 1304
1048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
104C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1054	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1054	DMA_TCD2_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1056	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	0000h	36.4.32/ 1311
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
105E	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	0000h	36.4.36/ 1314
1060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1064	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	0000h	36.4.23/ 1303
1066	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	0000h	36.4.24/ 1304
1068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
106C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1074	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1074	DMA_TCD3_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1076	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	0000h	36.4.32/ 1311
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
107E	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	0000h	36.4.36/ 1314
1080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1084	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	0000h	36.4.23/ 1303
1086	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	0000h	36.4.24/ 1304
1088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
108C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1094	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1094	DMA_TCD4_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1096	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	0000h	36.4.32/ 1311
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
109E	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	0000h	36.4.36/ 1314
10A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
10A4	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	0000h	36.4.23/ 1303
10A6	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	0000h	36.4.24/ 1304
10A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
10AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
10B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
10B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
10B4	DMA_TCD5_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10B6	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	0000h	36.4.32/ 1311
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
10BE	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	0000h	36.4.36/ 1314
10C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
10C4	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	0000h	36.4.23/ 1303
10C6	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	0000h	36.4.24/ 1304
10C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
10CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
10D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
10D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
10D4	DMA_TCD6_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
10D6	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	0000h	36.4.32/ 1311
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
10DE	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	0000h	36.4.36/ 1314
10E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10E4	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	0000h	36.4.23/ 1303
10E6	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	0000h	36.4.24/ 1304
10E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
10EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
10F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
10F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
10F4	DMA_TCD7_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
10F6	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	0000h	36.4.32/ 1311
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
10FE	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	0000h	36.4.36/ 1314
1100	TCD Source Address (DMA_TCD8_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1104	TCD Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	0000h	36.4.23/ 1303
1106	TCD Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	0000h	36.4.24/ 1304
1108	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
110C	TCD Last Source Address Adjustment (DMA_TCD8_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1110	TCD Destination Address (DMA_TCD8_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1114	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1114	DMA_TCD8_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1116	TCD Signed Destination Address Offset (DMA_TCD8_DOFF)	16	R/W	0000h	36.4.32/ 1311
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
111E	TCD Control and Status (DMA_TCD8_CSR)	16	R/W	0000h	36.4.36/ 1314
1120	TCD Source Address (DMA_TCD9_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1124	TCD Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	0000h	36.4.23/ 1303
1126	TCD Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	0000h	36.4.24/ 1304
1128	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
112C	TCD Last Source Address Adjustment (DMA_TCD9_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1130	TCD Destination Address (DMA_TCD9_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1134	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1134	DMA_TCD9_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1136	TCD Signed Destination Address Offset (DMA_TCD9_DOFF)	16	R/W	0000h	36.4.32/ 1311
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
113E	TCD Control and Status (DMA_TCD9_CSR)	16	R/W	0000h	36.4.36/ 1314
1140	TCD Source Address (DMA_TCD10_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1144	TCD Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	0000h	36.4.23/ 1303
1146	TCD Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	0000h	36.4.24/ 1304
1148	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
114C	TCD Last Source Address Adjustment (DMA_TCD10_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1150	TCD Destination Address (DMA_TCD10_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1154	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1154	DMA_TCD10_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1156	TCD Signed Destination Address Offset (DMA_TCD10_DOFF)	16	R/W	0000h	36.4.32/ 1311
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
115E	TCD Control and Status (DMA_TCD10_CSR)	16	R/W	0000h	36.4.36/ 1314
1160	TCD Source Address (DMA_TCD11_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1164	TCD Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	0000h	36.4.23/ 1303
1166	TCD Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	0000h	36.4.24/ 1304
1168	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
116C	TCD Last Source Address Adjustment (DMA_TCD11_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1170	TCD Destination Address (DMA_TCD11_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1174	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1174	DMA_TCD11_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1176	TCD Signed Destination Address Offset (DMA_TCD11_DOFF)	16	R/W	0000h	36.4.32/ 1311
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
117E	TCD Control and Status (DMA_TCD11_CSR)	16	R/W	0000h	36.4.36/ 1314
1180	TCD Source Address (DMA_TCD12_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1184	TCD Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	0000h	36.4.23/ 1303
1186	TCD Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	0000h	36.4.24/ 1304
1188	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
118C	TCD Last Source Address Adjustment (DMA_TCD12_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1190	TCD Destination Address (DMA_TCD12_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1194	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1194	DMA_TCD12_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1196	TCD Signed Destination Address Offset (DMA_TCD12_DOFF)	16	R/W	0000h	36.4.32/ 1311
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
119E	TCD Control and Status (DMA_TCD12_CSR)	16	R/W	0000h	36.4.36/ 1314
11A0	TCD Source Address (DMA_TCD13_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
11A4	TCD Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	0000h	36.4.23/ 1303
11A6	TCD Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	0000h	36.4.24/ 1304
11A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
11AC	TCD Last Source Address Adjustment (DMA_TCD13_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
11B0	TCD Destination Address (DMA_TCD13_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
11B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
11B4	DMA_TCD13_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
11B6	TCD Signed Destination Address Offset (DMA_TCD13_DOFF)	16	R/W	0000h	36.4.32/ 1311
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
11BE	TCD Control and Status (DMA_TCD13_CSR)	16	R/W	0000h	36.4.36/ 1314

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C0	TCD Source Address (DMA_TCD14_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
11C4	TCD Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	0000h	36.4.23/ 1303
11C6	TCD Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	0000h	36.4.24/ 1304
11C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
11CC	TCD Last Source Address Adjustment (DMA_TCD14_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
11D0	TCD Destination Address (DMA_TCD14_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
11D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
11D4	DMA_TCD14_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
11D6	TCD Signed Destination Address Offset (DMA_TCD14_DOFF)	16	R/W	0000h	36.4.32/ 1311
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
11DE	TCD Control and Status (DMA_TCD14_CSR)	16	R/W	0000h	36.4.36/ 1314
11E0	TCD Source Address (DMA_TCD15_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
11E4	TCD Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	0000h	36.4.23/ 1303
11E6	TCD Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	0000h	36.4.24/ 1304
11E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11EC	TCD Last Source Address Adjustment (DMA_TCD15_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
11F0	TCD Destination Address (DMA_TCD15_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
11F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
11F4	DMA_TCD15_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
11F6	TCD Signed Destination Address Offset (DMA_TCD15_DOFF)	16	R/W	0000h	36.4.32/ 1311
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
11FE	TCD Control and Status (DMA_TCD15_CSR)	16	R/W	0000h	36.4.36/ 1314
1200	TCD Source Address (DMA_TCD16_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1204	TCD Transfer Attributes (DMA_TCD16_ATTR)	16	R/W	0000h	36.4.23/ 1303
1206	TCD Signed Source Address Offset (DMA_TCD16_SOFF)	16	R/W	0000h	36.4.24/ 1304
1208	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD16_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD16_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD16_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
120C	TCD Last Source Address Adjustment (DMA_TCD16_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1210	TCD Destination Address (DMA_TCD16_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1214	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1214	DMA_TCD16_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1216	TCD Signed Destination Address Offset (DMA_TCD16_DOFF)	16	R/W	0000h	36.4.32/ 1311
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD16_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD16_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
121E	TCD Control and Status (DMA_TCD16_CSR)	16	R/W	0000h	36.4.36/ 1314
1220	TCD Source Address (DMA_TCD17_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1224	TCD Transfer Attributes (DMA_TCD17_ATTR)	16	R/W	0000h	36.4.23/ 1303
1226	TCD Signed Source Address Offset (DMA_TCD17_SOFF)	16	R/W	0000h	36.4.24/ 1304
1228	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD17_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD17_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD17_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
122C	TCD Last Source Address Adjustment (DMA_TCD17_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1230	TCD Destination Address (DMA_TCD17_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1234	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1234	DMA_TCD17_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1236	TCD Signed Destination Address Offset (DMA_TCD17_DOFF)	16	R/W	0000h	36.4.32/ 1311
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD17_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD17_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
123E	TCD Control and Status (DMA_TCD17_CSR)	16	R/W	0000h	36.4.36/ 1314
1240	TCD Source Address (DMA_TCD18_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1244	TCD Transfer Attributes (DMA_TCD18_ATTR)	16	R/W	0000h	36.4.23/ 1303

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1246	TCD Signed Source Address Offset (DMA_TCD18_SOFF)	16	R/W	0000h	36.4.24/ 1304
1248	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD18_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD18_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD18_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
124C	TCD Last Source Address Adjustment (DMA_TCD18_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1250	TCD Destination Address (DMA_TCD18_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1254	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1254	DMA_TCD18_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1256	TCD Signed Destination Address Offset (DMA_TCD18_DOFF)	16	R/W	0000h	36.4.32/ 1311
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD18_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD18_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
125E	TCD Control and Status (DMA_TCD18_CSR)	16	R/W	0000h	36.4.36/ 1314
1260	TCD Source Address (DMA_TCD19_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1264	TCD Transfer Attributes (DMA_TCD19_ATTR)	16	R/W	0000h	36.4.23/ 1303
1266	TCD Signed Source Address Offset (DMA_TCD19_SOFF)	16	R/W	0000h	36.4.24/ 1304
1268	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD19_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD19_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD19_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
126C	TCD Last Source Address Adjustment (DMA_TCD19_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1270	TCD Destination Address (DMA_TCD19_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1274	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1274	DMA_TCD19_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1276	TCD Signed Destination Address Offset (DMA_TCD19_DOFF)	16	R/W	0000h	36.4.32/ 1311
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD19_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD19_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
127E	TCD Control and Status (DMA_TCD19_CSR)	16	R/W	0000h	36.4.36/ 1314
1280	TCD Source Address (DMA_TCD20_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1284	TCD Transfer Attributes (DMA_TCD20_ATTR)	16	R/W	0000h	36.4.23/ 1303
1286	TCD Signed Source Address Offset (DMA_TCD20_SOFF)	16	R/W	0000h	36.4.24/ 1304
1288	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD20_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD20_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD20_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
128C	TCD Last Source Address Adjustment (DMA_TCD20_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1290	TCD Destination Address (DMA_TCD20_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1294	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1294	DMA_TCD20_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1296	TCD Signed Destination Address Offset (DMA_TCD20_DOFF)	16	R/W	0000h	36.4.32/ 1311
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD20_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD20_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
129E	TCD Control and Status (DMA_TCD20_CSR)	16	R/W	0000h	36.4.36/ 1314
12A0	TCD Source Address (DMA_TCD21_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
12A4	TCD Transfer Attributes (DMA_TCD21_ATTR)	16	R/W	0000h	36.4.23/ 1303
12A6	TCD Signed Source Address Offset (DMA_TCD21_SOFF)	16	R/W	0000h	36.4.24/ 1304
12A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD21_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD21_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD21_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
12AC	TCD Last Source Address Adjustment (DMA_TCD21_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
12B0	TCD Destination Address (DMA_TCD21_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
12B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
12B4	DMA_TCD21_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
12B6	TCD Signed Destination Address Offset (DMA_TCD21_DOFF)	16	R/W	0000h	36.4.32/ 1311
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD21_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD21_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
12BE	TCD Control and Status (DMA_TCD21_CSR)	16	R/W	0000h	36.4.36/ 1314
12C0	TCD Source Address (DMA_TCD22_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
12C4	TCD Transfer Attributes (DMA_TCD22_ATTR)	16	R/W	0000h	36.4.23/ 1303
12C6	TCD Signed Source Address Offset (DMA_TCD22_SOFF)	16	R/W	0000h	36.4.24/ 1304
12C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD22_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD22_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD22_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
12CC	TCD Last Source Address Adjustment (DMA_TCD22_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
12D0	TCD Destination Address (DMA_TCD22_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
12D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
12D4	DMA_TCD22_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
12D6	TCD Signed Destination Address Offset (DMA_TCD22_DOFF)	16	R/W	0000h	36.4.32/ 1311
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD22_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD22_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
12DE	TCD Control and Status (DMA_TCD22_CSR)	16	R/W	0000h	36.4.36/ 1314
12E0	TCD Source Address (DMA_TCD23_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
12E4	TCD Transfer Attributes (DMA_TCD23_ATTR)	16	R/W	0000h	36.4.23/ 1303
12E6	TCD Signed Source Address Offset (DMA_TCD23_SOFF)	16	R/W	0000h	36.4.24/ 1304
12E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD23_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD23_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD23_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
12EC	TCD Last Source Address Adjustment (DMA_TCD23_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
12F0	TCD Destination Address (DMA_TCD23_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
12F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
12F4	DMA_TCD23_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12F6	TCD Signed Destination Address Offset (DMA_TCD23_DOFF)	16	R/W	0000h	36.4.32/ 1311
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD23_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD23_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
12FE	TCD Control and Status (DMA_TCD23_CSR)	16	R/W	0000h	36.4.36/ 1314
1300	TCD Source Address (DMA_TCD24_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1304	TCD Transfer Attributes (DMA_TCD24_ATTR)	16	R/W	0000h	36.4.23/ 1303
1306	TCD Signed Source Address Offset (DMA_TCD24_SOFF)	16	R/W	0000h	36.4.24/ 1304
1308	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD24_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD24_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD24_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
130C	TCD Last Source Address Adjustment (DMA_TCD24_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1310	TCD Destination Address (DMA_TCD24_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1314	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1314	DMA_TCD24_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1316	TCD Signed Destination Address Offset (DMA_TCD24_DOFF)	16	R/W	0000h	36.4.32/ 1311
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD24_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD24_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
131E	TCD Control and Status (DMA_TCD24_CSR)	16	R/W	0000h	36.4.36/ 1314

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1320	TCD Source Address (DMA_TCD25_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1324	TCD Transfer Attributes (DMA_TCD25_ATTR)	16	R/W	0000h	36.4.23/ 1303
1326	TCD Signed Source Address Offset (DMA_TCD25_SOFF)	16	R/W	0000h	36.4.24/ 1304
1328	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD25_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD25_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD25_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
132C	TCD Last Source Address Adjustment (DMA_TCD25_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1330	TCD Destination Address (DMA_TCD25_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1334	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1334	DMA_TCD25_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1336	TCD Signed Destination Address Offset (DMA_TCD25_DOFF)	16	R/W	0000h	36.4.32/ 1311
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD25_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD25_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
133E	TCD Control and Status (DMA_TCD25_CSR)	16	R/W	0000h	36.4.36/ 1314
1340	TCD Source Address (DMA_TCD26_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1344	TCD Transfer Attributes (DMA_TCD26_ATTR)	16	R/W	0000h	36.4.23/ 1303
1346	TCD Signed Source Address Offset (DMA_TCD26_SOFF)	16	R/W	0000h	36.4.24/ 1304
1348	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD26_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD26_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD26_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
134C	TCD Last Source Address Adjustment (DMA_TCD26_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1350	TCD Destination Address (DMA_TCD26_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1354	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1354	DMA_TCD26_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1356	TCD Signed Destination Address Offset (DMA_TCD26_DOFF)	16	R/W	0000h	36.4.32/ 1311
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD26_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD26_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
135E	TCD Control and Status (DMA_TCD26_CSR)	16	R/W	0000h	36.4.36/ 1314
1360	TCD Source Address (DMA_TCD27_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1364	TCD Transfer Attributes (DMA_TCD27_ATTR)	16	R/W	0000h	36.4.23/ 1303
1366	TCD Signed Source Address Offset (DMA_TCD27_SOFF)	16	R/W	0000h	36.4.24/ 1304
1368	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD27_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD27_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD27_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
136C	TCD Last Source Address Adjustment (DMA_TCD27_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1370	TCD Destination Address (DMA_TCD27_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1374	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1374	DMA_TCD27_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1376	TCD Signed Destination Address Offset (DMA_TCD27_DOFF)	16	R/W	0000h	36.4.32/ 1311
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD27_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD27_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
137E	TCD Control and Status (DMA_TCD27_CSR)	16	R/W	0000h	36.4.36/ 1314
1380	TCD Source Address (DMA_TCD28_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
1384	TCD Transfer Attributes (DMA_TCD28_ATTR)	16	R/W	0000h	36.4.23/ 1303
1386	TCD Signed Source Address Offset (DMA_TCD28_SOFF)	16	R/W	0000h	36.4.24/ 1304
1388	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD28_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD28_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD28_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
138C	TCD Last Source Address Adjustment (DMA_TCD28_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
1390	TCD Destination Address (DMA_TCD28_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
1394	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
1394	DMA_TCD28_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
1396	TCD Signed Destination Address Offset (DMA_TCD28_DOFF)	16	R/W	0000h	36.4.32/ 1311
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD28_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD28_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
139E	TCD Control and Status (DMA_TCD28_CSR)	16	R/W	0000h	36.4.36/ 1314
13A0	TCD Source Address (DMA_TCD29_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
13A4	TCD Transfer Attributes (DMA_TCD29_ATTR)	16	R/W	0000h	36.4.23/ 1303

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13A6	TCD Signed Source Address Offset (DMA_TCD29_SOFF)	16	R/W	0000h	36.4.24/ 1304
13A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD29_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD29_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD29_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
13AC	TCD Last Source Address Adjustment (DMA_TCD29_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
13B0	TCD Destination Address (DMA_TCD29_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
13B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
13B4	DMA_TCD29_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
13B6	TCD Signed Destination Address Offset (DMA_TCD29_DOFF)	16	R/W	0000h	36.4.32/ 1311
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD29_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD29_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
13BE	TCD Control and Status (DMA_TCD29_CSR)	16	R/W	0000h	36.4.36/ 1314
13C0	TCD Source Address (DMA_TCD30_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
13C4	TCD Transfer Attributes (DMA_TCD30_ATTR)	16	R/W	0000h	36.4.23/ 1303
13C6	TCD Signed Source Address Offset (DMA_TCD30_SOFF)	16	R/W	0000h	36.4.24/ 1304
13C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD30_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD30_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD30_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
13CC	TCD Last Source Address Adjustment (DMA_TCD30_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
13D0	TCD Destination Address (DMA_TCD30_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
13D4	DMA_TCD30_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
13D6	TCD Signed Destination Address Offset (DMA_TCD30_DOFF)	16	R/W	0000h	36.4.32/ 1311
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD30_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD30_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
13DE	TCD Control and Status (DMA_TCD30_CSR)	16	R/W	0000h	36.4.36/ 1314
13E0	TCD Source Address (DMA_TCD31_SADDR)	32	R/W	0000_0000h	36.4.22/ 1303
13E4	TCD Transfer Attributes (DMA_TCD31_ATTR)	16	R/W	0000h	36.4.23/ 1303
13E6	TCD Signed Source Address Offset (DMA_TCD31_SOFF)	16	R/W	0000h	36.4.24/ 1304
13E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD31_NBYTES_MLNO)	32	R/W	0000_0000h	36.4.25/ 1305
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD31_NBYTES_MLOFFNO)	32	R/W	0000_0000h	36.4.26/ 1305
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD31_NBYTES_MLOFFYES)	32	R/W	0000_0000h	36.4.27/ 1307
13EC	TCD Last Source Address Adjustment (DMA_TCD31_SLAST)	32	R/W	0000_0000h	36.4.28/ 1308
13F0	TCD Destination Address (DMA_TCD31_DADDR)	32	R/W	0000_0000h	36.4.29/ 1308
13F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_CITER_ELINKYES)	16	R/W	0000h	36.4.30/ 1309
13F4	DMA_TCD31_CITER_ELINKNO	16	R/W	0000h	36.4.31/ 1310
13F6	TCD Signed Destination Address Offset (DMA_TCD31_DOFF)	16	R/W	0000h	36.4.32/ 1311
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD31_DLASTSGA)	32	R/W	0000_0000h	36.4.33/ 1311
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_BITER_ELINKYES)	16	R/W	0000h	36.4.34/ 1312

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD31_BITER_ELINKNO)	16	R/W	0000h	36.4.35/ 1313
13FE	TCD Control and Status (DMA_TCD31_CSR)	16	R/W	0000h	36.4.36/ 1314

36.4.5 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA. The DMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels within each group are cycled through (from high to low channel number) without regard to priority.

NOTE

For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error will be reported. For group round robin arbitration, the group priorities are ignored and the groups are cycled through (from high to low group number) without regard to priority.

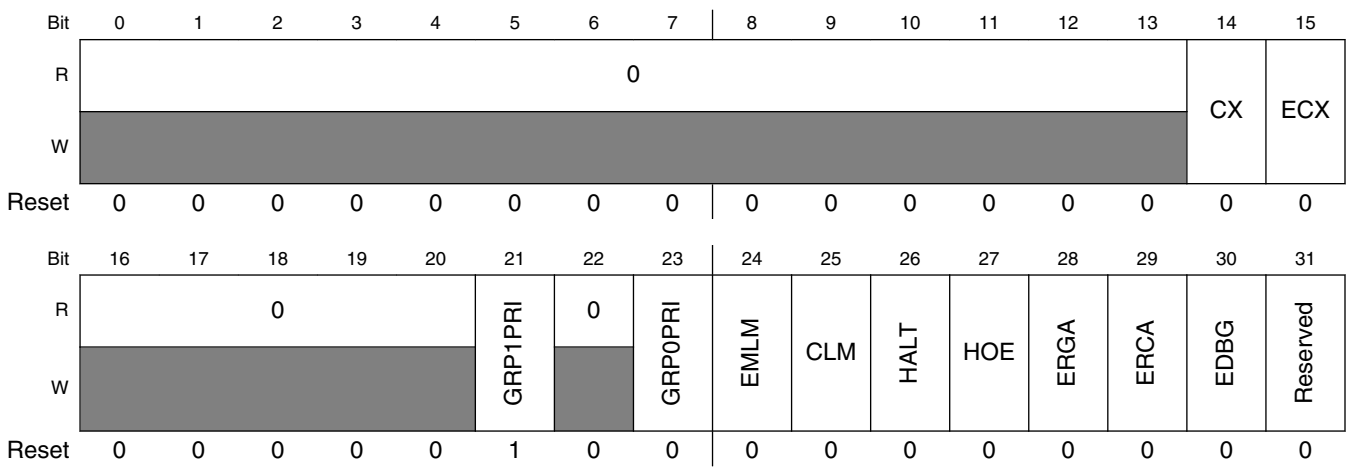
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is

complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 0h base + 0h offset = 0h



DMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
25 CLM	Continuous Link Mode **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
31 Reserved	This field is reserved. Reserved

36.4.6 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- The UCE bit shows an uncorrectable error occurred while the device has ECC protection on the TCD SRAM
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
0 VLD	Logical OR of all ERR status bits

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
	0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error during channel execution. The UCE bit is set when an uncorrectable ECC error occurs on an access generated by the DMA only. If a CPU access to the TCD causes an uncorrectable ECC error, that access will receive a bus error response. **NOTE:** When the eDMA sees a RAM error on an IPS access (when the user is accessing a TCD), the error is reported as a bus abort. When the dma_engine (the execution engine is accessing a TCD) receives a RAM error, it is recorded in the Error Status register, DMA_ES[UCE], along with the channel number. 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding GPE and CPE errors, or last recorded error canceled transfer.
24 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
25 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
26 DAE	Destination Address Error

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
	0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
29 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
30 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read
31 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

36.4.7 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQ field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

36.4.8 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEI field descriptions

Field	Description
0 EEI31	Enable Error Interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable Error Interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
4 EEI27	Enable Error Interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable Error Interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable Error Interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
16 EEI15	Enable Error Interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable Error Interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable Error Interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
28 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

36.4.9 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	SAER	0	SERQ				
Reset	0	0	0	0	0	0	0	0

DMA_SERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAER	Set All Enable Requests

Table continues on the next page...

DMA_SERQ field descriptions (continued)

Field	Description
	0 Set only the ERQ bit specified in the SERQ field 1 Set all bits in ERQ
2 Reserved	This field is reserved.
3–7 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

36.4.10 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 19h offset = 19h

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAER	0	CERQ				
Reset	0	0	0	0	0	0	0	0

DMA_CERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAER	Clear All Enable Requests 0 Clear only the ERQ bit specified in the CERQ field 1 Clear all bits in ERQ
2 Reserved	This field is reserved.
3–7 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

36.4.11 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	SAEE	0			SEEI		
Reset	0	0	0	0	0	0	0	0

DMA_SEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI bit specified in the SEEI field. 1 Sets all bits in EEI
2 Reserved	This field is reserved.
3–7 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

36.4.12 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Bh offset = 1Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	CAEE	0			CEEI		
Reset	0	0	0	0	0	0	0	0

DMA_CEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
2 Reserved	This field is reserved.
3–7 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

36.4.13 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	CAIR	0			CINT		
Reset	0	0	0	0	0	0	0	0

DMA_CINT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAIR	Clear All Interrupt Requests 0 Clear only the INT bit specified in the CINT field 1 Clear all bits in INT
2 Reserved	This field is reserved.
3–7 CINT	Clear Interrupt Request Clears the corresponding bit in INT

36.4.14 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Dh offset = 1Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

DMA_CERR field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEI	Clear All Error Indicators 0 Clear only the ERR bit specified in the CERR field 1 Clear all bits in ERR
2 Reserved	This field is reserved.
3–7 CERR	Clear Error Indicator Clears the corresponding bit in ERR

36.4.15 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	SAST	0			SSRT		
Reset	0	0	0	0	0	0	0	0

DMA_SSRT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
2 Reserved	This field is reserved.
3–7 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

36.4.16 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Fh offset = 1Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CADN	0	CDNE				
Reset	0	0	0	0	0	0	0	0

DMA_CDNE field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
2 Reserved	This field is reserved.
3–7 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

36.4.17 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INT field descriptions

Field	Description
0 INT31	Interrupt Request 31

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt Request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

36.4.18 Error Register (DMA_ERR)

The ERR provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERR field descriptions

Field	Description
0 ERR31	Error In Channel 31 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR30	Error In Channel 30 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR29	Error In Channel 29 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR28	Error In Channel 28 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
4 ERR27	Error In Channel 27 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR26	Error In Channel 26 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR25	Error In Channel 25 0 An error in this channel has not occurred 1 An error in this channel has occurred
7 ERR24	Error In Channel 24 0 An error in this channel has not occurred 1 An error in this channel has occurred
8 ERR23	Error In Channel 23 0 An error in this channel has not occurred 1 An error in this channel has occurred
9 ERR22	Error In Channel 22 0 An error in this channel has not occurred 1 An error in this channel has occurred
10 ERR21	Error In Channel 21 0 An error in this channel has not occurred 1 An error in this channel has occurred
11 ERR20	Error In Channel 20 0 An error in this channel has not occurred 1 An error in this channel has occurred
12 ERR19	Error In Channel 19 0 An error in this channel has not occurred 1 An error in this channel has occurred
13 ERR18	Error In Channel 18 0 An error in this channel has not occurred 1 An error in this channel has occurred
14 ERR17	Error In Channel 17 0 An error in this channel has not occurred 1 An error in this channel has occurred
15 ERR16	Error In Channel 16 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
16 ERR15	Error In Channel 15 0 An error in this channel has not occurred 1 An error in this channel has occurred
17 ERR14	Error In Channel 14 0 An error in this channel has not occurred 1 An error in this channel has occurred
18 ERR13	Error In Channel 13 0 An error in this channel has not occurred 1 An error in this channel has occurred
19 ERR12	Error In Channel 12 0 An error in this channel has not occurred 1 An error in this channel has occurred
20 ERR11	Error In Channel 11 0 An error in this channel has not occurred 1 An error in this channel has occurred
21 ERR10	Error In Channel 10 0 An error in this channel has not occurred 1 An error in this channel has occurred
22 ERR9	Error In Channel 9 0 An error in this channel has not occurred 1 An error in this channel has occurred
23 ERR8	Error In Channel 8 0 An error in this channel has not occurred 1 An error in this channel has occurred
24 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
25 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
26 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
27 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
28 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
29 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
30 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
31 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

36.4.19 Hardware Request Status Register (DMA_HRS)

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRS field descriptions

Field	Description
0 HRS31	Hardware Request Status Channel 31 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 31 is not present 1 A hardware service request for channel 31 is present
1 HRS30	Hardware Request Status Channel 30 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 30 is not present 1 A hardware service request for for channel 30 is present
2 HRS29	Hardware Request Status Channel 29 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 29 is not preset 1 A hardware service request for channel 29 is present
3 HRS28	Hardware Request Status Channel 28 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 28 is not present 1 A hardware service request for channel 28 is present
4 HRS27	Hardware Request Status Channel 27 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 27 is not present 1 A hardware service request for channel 27 is present
5 HRS26	Hardware Request Status Channel 26 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 26 is not present 1 A hardware service request for channel 26 is present
6 HRS25	Hardware Request Status Channel 25 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 25 is not present 1 A hardware service request for channel 25 is present
7 HRS24	Hardware Request Status Channel 24 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 24 is not present 1 A hardware service request for channel 24 is present
8 HRS23	Hardware Request Status Channel 23 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 23 is not present 1 A hardware service request for channel 23 is present
9 HRS22	Hardware Request Status Channel 22 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 22 is not present 1 A hardware service request for channel 22 is present
10 HRS21	Hardware Request Status Channel 21 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 21 is not present 1 A hardware service request for channel 21 is present
11 HRS20	Hardware Request Status Channel 20

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 20 is not present 1 A hardware service request for channel 20 is present
12 HRS19	Hardware Request Status Channel 19 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 19 is not present 1 A hardware service request for channel 19 is present
13 HRS18	Hardware Request Status Channel 18 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 18 is not present 1 A hardware service request for channel 18 is present
14 HRS17	Hardware Request Status Channel 17 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 17 is not present 1 A hardware service request for channel 17 is present
15 HRS16	Hardware Request Status Channel 16 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 16 is not present 1 A hardware service request for channel 16 is present
16 HRS15	Hardware Request Status Channel 15 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 15 is not present 1 A hardware service request for channel 15 is present
17 HRS14	Hardware Request Status Channel 14 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 14 is not present 1 A hardware service request for channel 14 is present
18 HRS13	Hardware Request Status Channel 13 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 13 is not present 1 A hardware service request for channel 13 is present
19 HRS12	Hardware Request Status Channel 12 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 12 is not present 1 A hardware service request for channel 12 is present
20 HRS11	Hardware Request Status Channel 11 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 11 is not present 1 A hardware service request for channel 11 is present
21 HRS10	Hardware Request Status Channel 10 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 10 is not present 1 A hardware service request for channel 10 is present
22 HRS9	Hardware Request Status Channel 9 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 9 is not present 1 A hardware service request for channel 9 is present
23 HRS8	Hardware Request Status Channel 8 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 8 is not present 1 A hardware service request for channel 8 is present
24 HRS7	Hardware Request Status Channel 7

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 7 is not present 1 A hardware service request for channel 7 is present
25 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 6 is not present 1 A hardware service request for channel 6 is present
26 HRS5	Hardware Request Status Channel 5 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 5 is not present 1 A hardware service request for channel 5 is present
27 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 4 is not present 1 A hardware service request for channel 4 is present
28 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 3 is not present 1 A hardware service request for channel 3 is present
29 HRS2	Hardware Request Status Channel 2 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 2 is not present 1 A hardware service request for channel 2 is present
30 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 1 is not present 1 A hardware service request for channel 1 is present
31 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 0 is not present 1 A hardware service request for channel 0 is present

36.4.20 Channel n Priority Register (DMA_DCHPRI*n*)

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

Address: 0h base + 100h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ECP	DPA	GRPPRI		CHPRI			
Write								
Reset	0	0	*	*	*	*	*	*

* Notes:
- CHPRI field: See bit field description.
- GRPPRI field: See bit field description.

DMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel's service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.

Table continues on the next page...

DMA_DCHPRI*n* field descriptions (continued)

Field	Description
1 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
2–3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored. **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4–7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] = 0b01111.

36.4.21 Channel n Master ID Register (DMA_DCHMID*n*)

The DMA master ID replication registers allow the DMA to use the same protection level and system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master, for example, a core, programs a TCD, its master ID and protection level are captured when the TCDn.CSR control attributes are written. Although the scatter/gather operation can change the contents of TCDn_CSR, that operation does not affect the DCHMID *n* registers.

Address: 0h base + 140h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	EMI	PAL	0		MID			
Write								
Reset	0	0	0	0	0	0	0	0

DMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication **NOTE:** If Master ID replication is disabled, the privileged protection level (supervisor mode) for DMA transfers is used.

Table continues on the next page...

DMA_DCHMID*n* field descriptions (continued)

Field	Description
	0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level **NOTE:** The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCDn.CSR. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MID	Master ID DMA's master ID when channel n is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCDn.CSR.

36.4.22 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 0h base + 1000h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SADDR field descriptions

Field	Description
0–31 SADDR	Source Address Memory address pointing to the source data.

36.4.23 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 0h base + 1004h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMOD					SSIZE			DMOD					DSIZE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_ATTR field descriptions

Field	Description
0–4 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst (4 beats of 64 bits) 110 Reserved 111 Reserved
8–12 DMOD	Destination Address Modulo See the SMOD definition
13–15 DSIZE	Destination data transfer size See the SSIZE definition

36.4.24 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 0h base + 1006h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SOFF field descriptions

Field	Description
0–15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

36.4.25 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

36.4.26 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

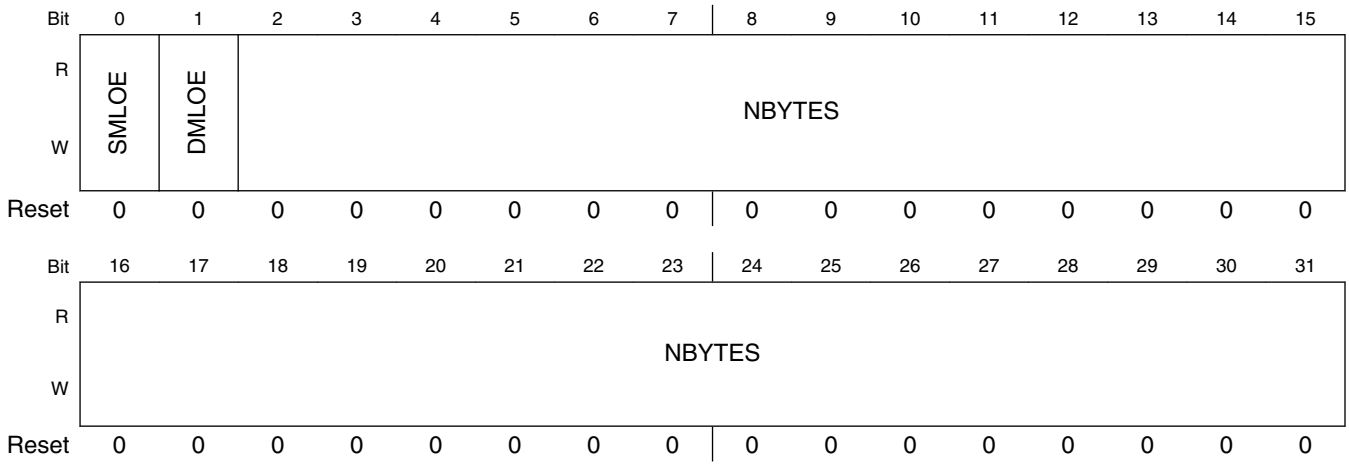
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d



DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

36.4.27 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

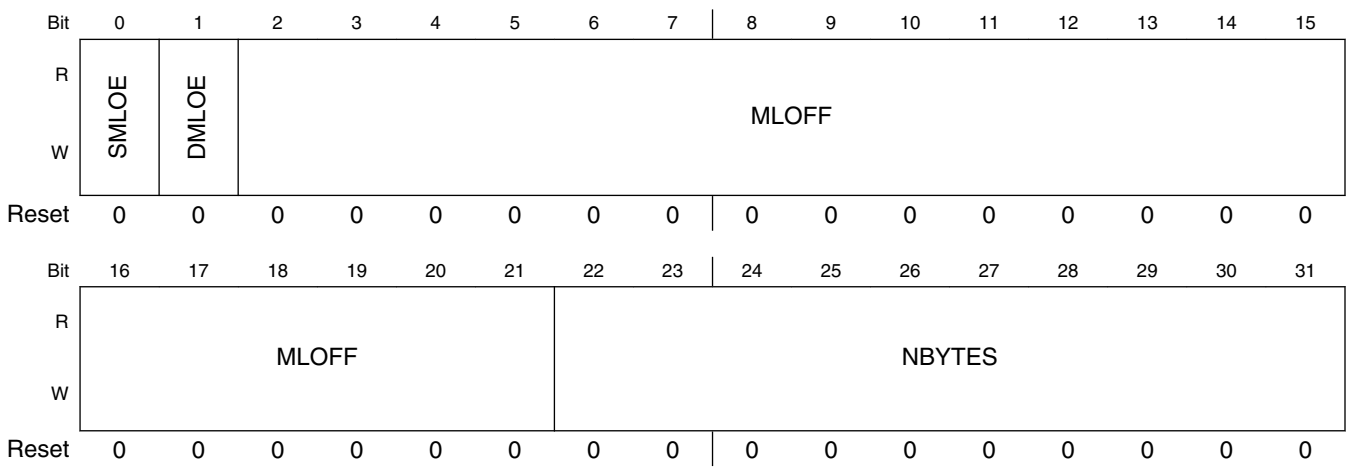
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d



DMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR

Table continues on the next page...

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions (continued)

Field	Description
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

36.4.28 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 0h base + 100Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SLAST field descriptions

Field	Description
0–31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

36.4.29 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 0h base + 1010h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DADDR field descriptions

Field	Description
0–31 DADDR	Destination Address Memory address pointing to the destination data.

36.4.30 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ELINK	LINKCH						CITER
Write		0						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CITER							
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

36.4.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	CITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

36.4.32 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 0h base + 1016h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DOFF field descriptions

Field	Description
0–15 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

36.4.33 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 0h base + 1018h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

36.4.34 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	0	LINKCH					BITER
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
7–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

36.4.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	BITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

36.4.36 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 0h base + 101Eh offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	BWC			MAJORLINKCH				
Write			0					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CSR field descriptions

Field	Description
0–1 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls. 01 Enable eDMA master high-priority elevation (HPE) mode. No eDMA engine stalls. 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
2 Reserved	This field is reserved.
3–7 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
8 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
9 ACTIVE	Channel Active

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
15 START	Channel Start

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

36.5 Functional description

The operation of the eDMA is described in the following subsections.

36.5.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

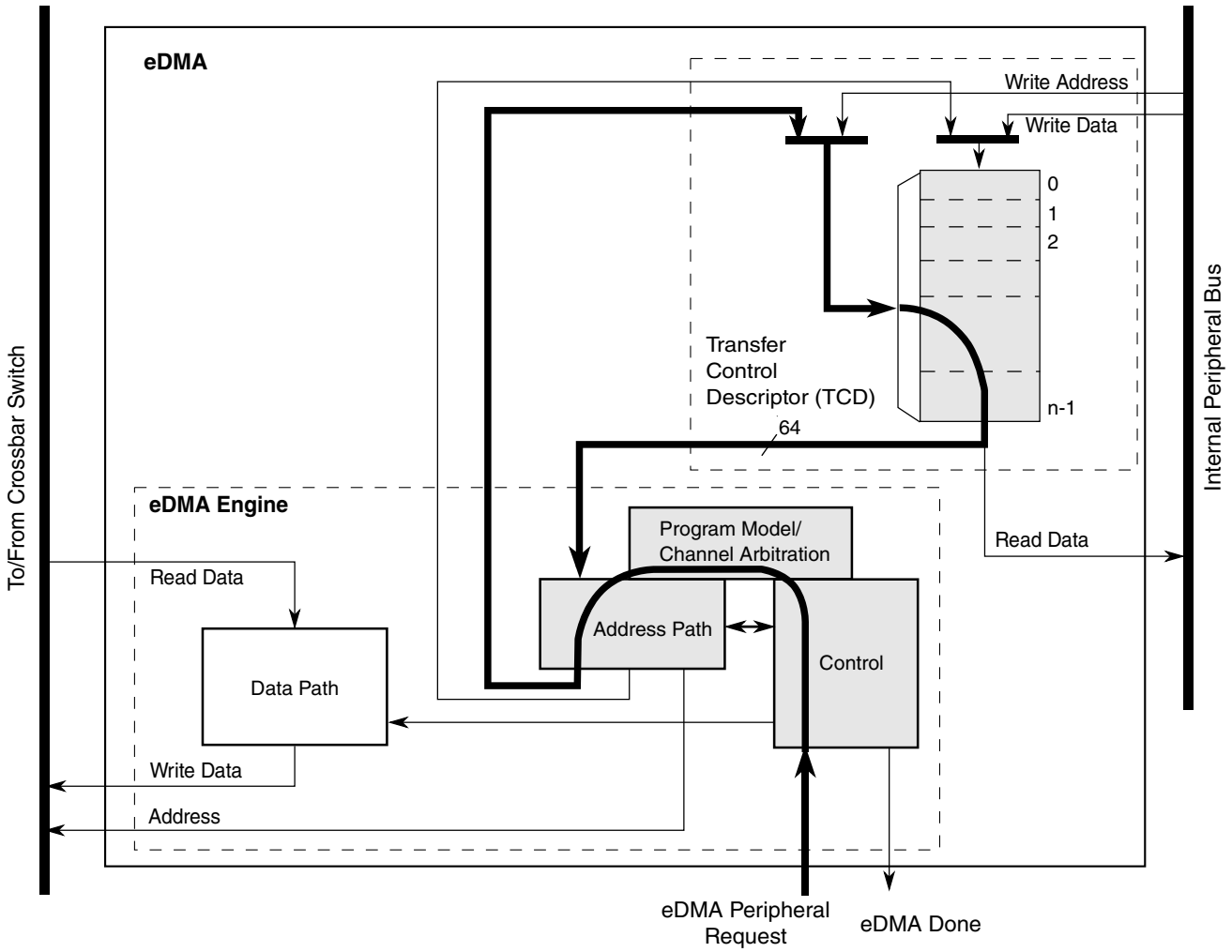


Figure 36-3. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

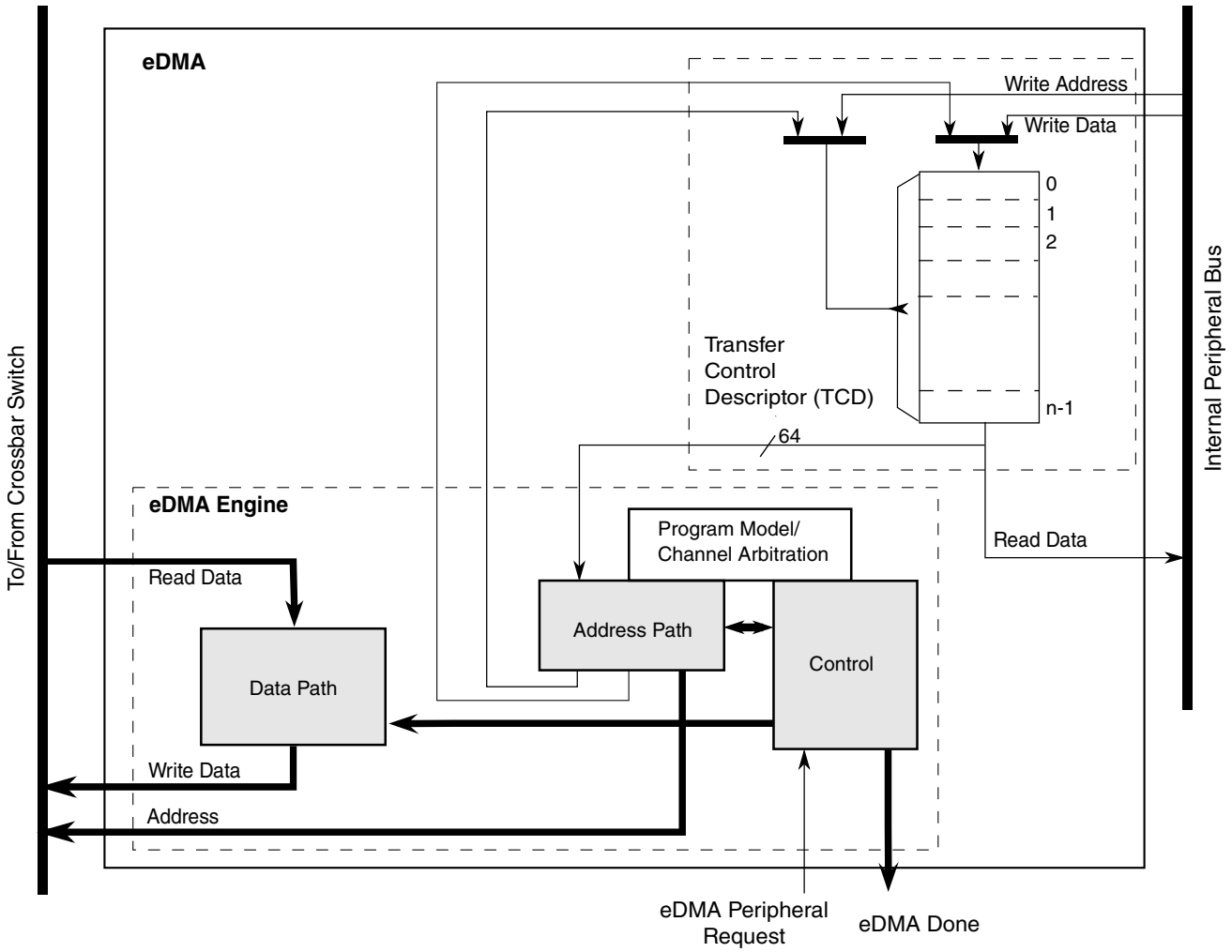


Figure 36-4. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

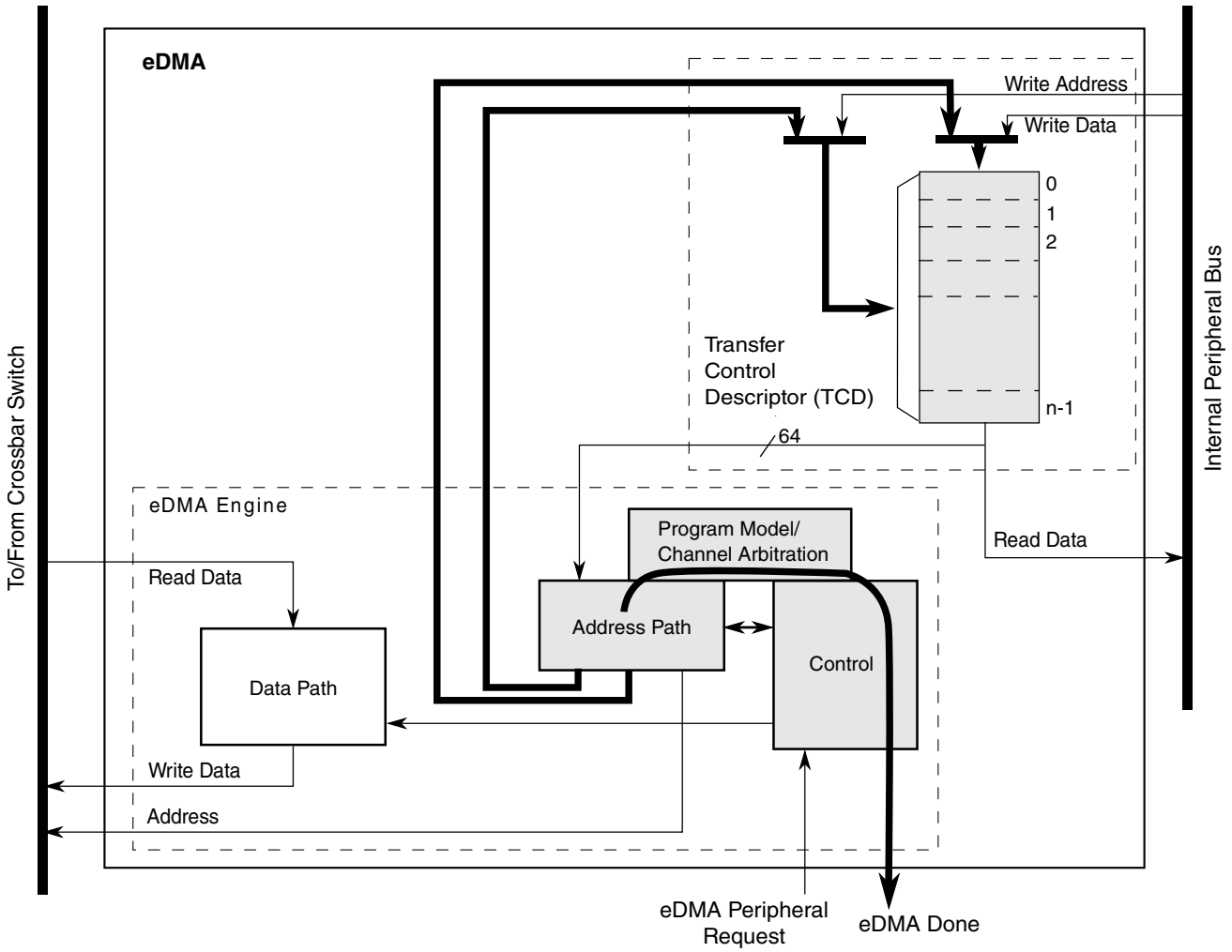


Figure 36-5. eDMA operation, part 3

36.5.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.

- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error

occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request,

are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

36.5.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

36.6 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

36.6.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.

2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 36-4. TCD Control and Status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

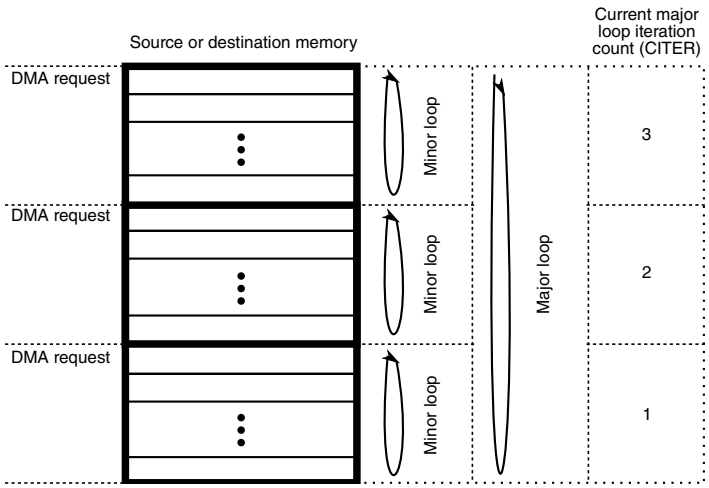


Figure 36-6. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

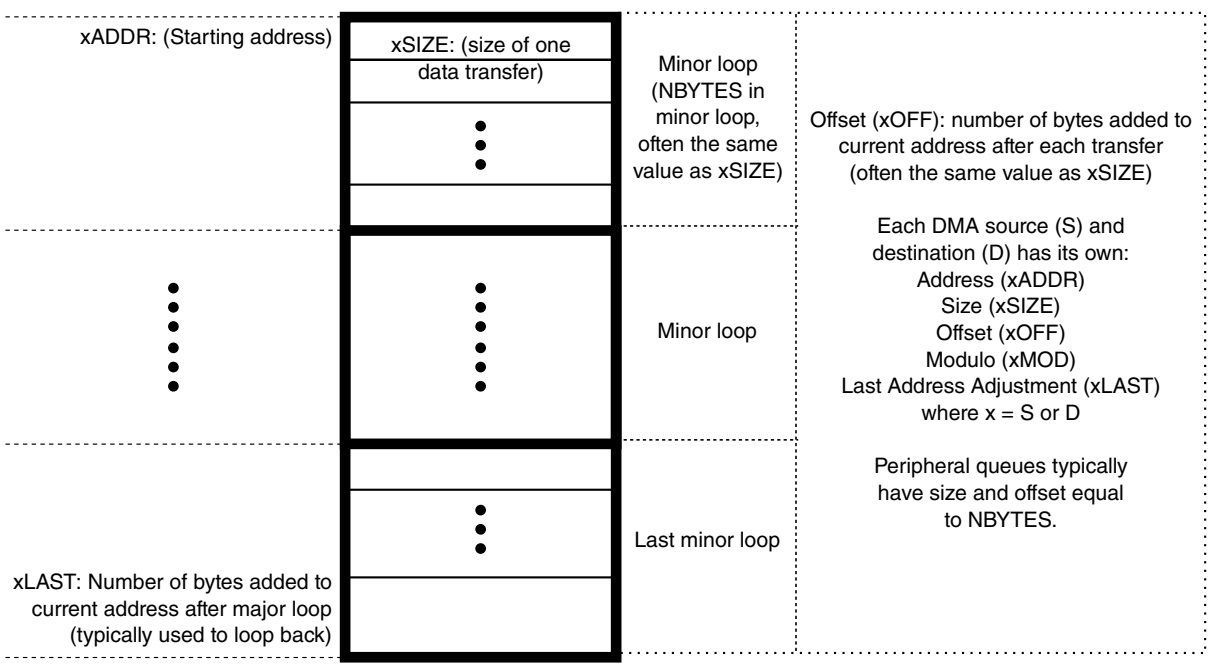


Figure 36-7. Memory array terms

36.6.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

36.6.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

36.6.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

36.6.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

36.6.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

36.6.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
    b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
    c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.

d. Write 32-bits to location 0x2004 → second iteration of the minor loop.

e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

36.6.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:

   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.

b. Write 32-bits to location 0x2000 → first iteration of the minor loop.

c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.

d. Write 32-bits to location 0x2004 → second iteration of the minor loop.

e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

b. Write 32-bits to location 0x2010 → first iteration of the minor loop.

c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

d. Write 32-bits to location 0x2014 → second iteration of the minor loop.

e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

f. Write 32-bits to location 0x2018 → third iteration of the minor loop.

g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).

15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

36.6.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 36-5. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

36.6.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

36.6.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

36.6.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

36.6.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

36.6.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
```

```
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 36-6. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

36.6.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

36.6.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

36.6.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   - If TCD.major.e_link = 1, the dynamic link attempt was successful.
   - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared

automatically by the eDMA engine after a channel begins execution.

36.6.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources.When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

36.6.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD indentification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.
6. Test the TCD.e_sg request status and TCD.major.linkch value:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

36.6.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD indentification (ID).

1. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

2. Write theTCD.dlast_sga field with the scatter/gather address.
3. Write 1b to the TCD.e_sg bit.
4. Read back the TCD.e_sg bit.

5. Test the TCD.e_sg request status:

   If e_sg = 1b, the dynamic link attempt was successful.

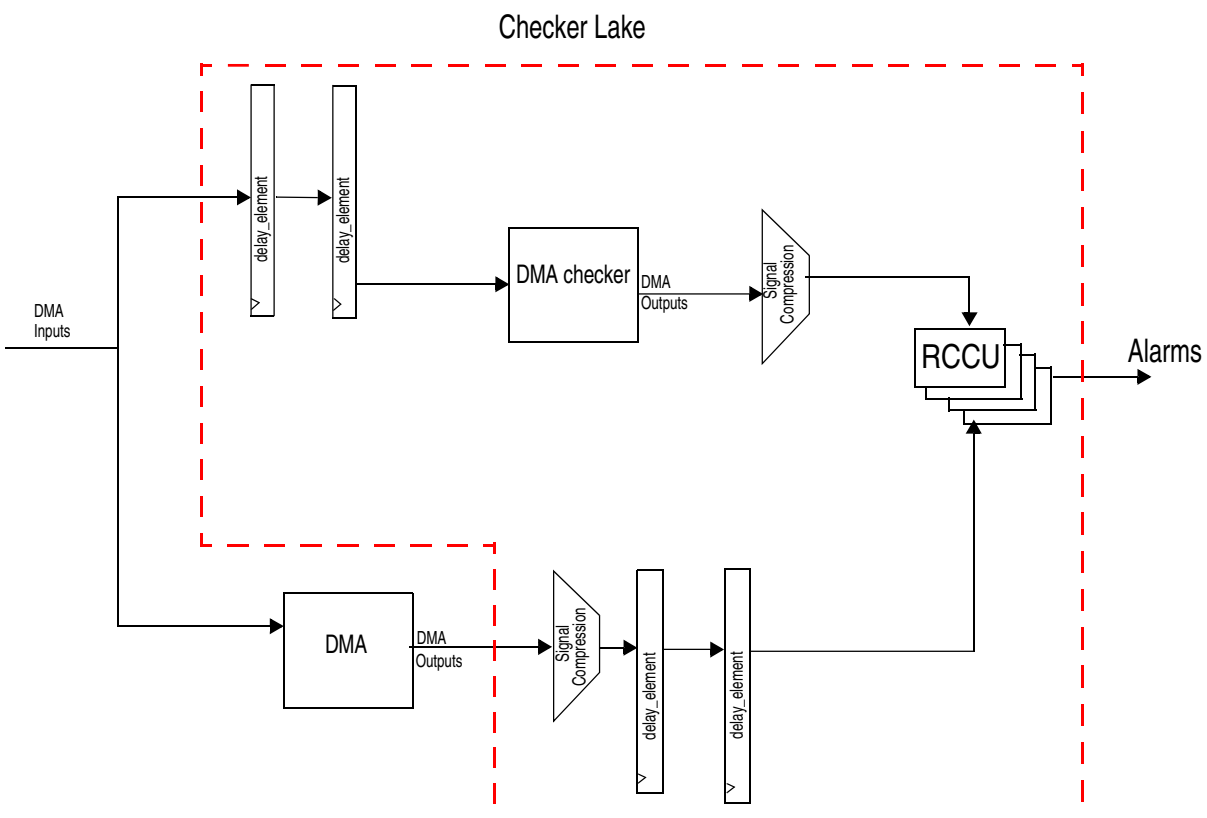
   If e_sg = 0b, read the 32 bit TCD dlast_sga field.

   If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

36.6.8 Lockstep

This chip contains an eDMA Checker which, when enabled, operates in Lockstep mode with the primary eDMA, executing the exact same transfer profiles. The eDMA Checker checks to ensure the primary eDMA is operating correctly. A block diagram is shown below.



36.6.8.1 Initialization

To prevent the eDMA Checker from generating a false comparison error after power-on-reset, the eDMA's data output buffer must be initialized prior to enabling the Checker. To initialize the eDMA data output buffer, the eDMA must perform a one or more 64-bit data transfers using any data value.

36.6.8.2 Errors

To handle source-address errors (SAE), perform the following steps.

1. Fix the addressing error.
2. Clear the error bit in Error Register (DMA_ERR).
3. Start the channel in a normal fashion.

NOTE

Error Status Register (DMA_ES) is read-only; the bits cannot be cleared. It saves the last recorded error. The VLD bit shows the user whether any error bits in the Error Register are set, thus indicating an error occurred that hasn't been cleared.

Chapter 37
Quad-ported RAM controller (PRAMC)

37.1 Introduction

This section provides an overview of the quad-ported Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and multiple banks of integrated system RAM, enabling simultaneous SRAM accesses by multiple bus masters.

Internally the quad-ported RAM controller provides an independently-configurable RAM controller for each associated bank of SRAM. One of the internal RAM controllers supports the use of its associated SRAM as overlay SRAM. See the chip-specific information for details.

Figure 37-1 illustrates the quad-port RAM controller in the context of a typical SoC platform architecture. The RAM controller supports four 64-bit primary AHB interfaces, each to a separate system crossbar switch slave port, a backdoor AHB port to the on-chip flash controller, and a 64-bit RAM array interface to each associated bank of on-chip SRAM.

The primary AHB ports provide a connection to the platform crossbar for direct RAM access from the various system crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access.

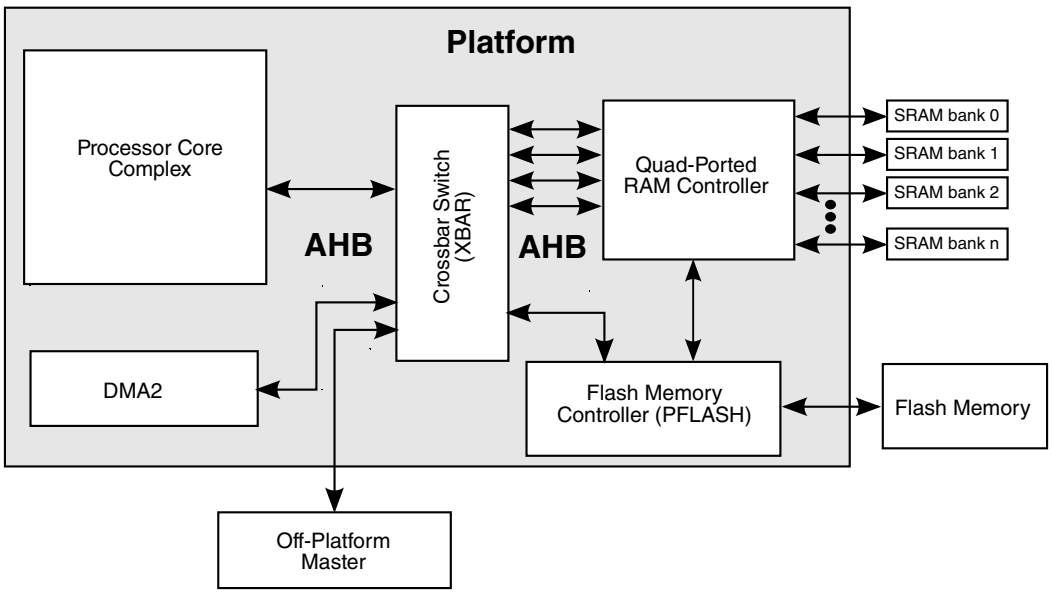


Figure 37-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- End-to-End ECC (e2eECC) support
- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interfaces support a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a double word

NOTE

1. Read-modify-write operations are performed for writes of less than 64-bits and writes not aligned with the 64-bit segments for which ECC check bits are calculated. The read-modify-write operations are necessary to preserve the integrity of the End-to_end ECC (e2eECC) check bits. See e2eECC considerations on less-than-64-bit write transactions for a detailed explanation.
2. The RAM controller supports parallel read-modify-write operations only to separate SRAM banks.

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write

request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB data phase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read data path contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read data bus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire data path of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write data path contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

37.2 SRAM controller memory map and register definition

The quad-ported RAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

1. Due to the structure of the quad-ported RAM controller, there are other programming considerations that are discussed in the PRAM_XBAR section of the Embedded Memories chapter.
2. Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory

references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0100h	37.2.1/1342
4	Platform RAM Configuration Register 2 (PRAMC_PRCR2)	32	R/W	0000_0100h	37.2.2/1343
8	Platform RAM Configuration Register 3 (PRAMC_PRCR3)	32	R/W	0000_0100h	37.2.3/1345
C	Platform RAM Configuration Register 4 (PRAMC_PRCR4)	32	R/W	0000_0100h	37.2.4/1346
10	Platform RAM Configuration Register 5 (PRAMC_PRCR5)	32	R/W	0000_0100h	37.2.5/1347
14	Platform RAM Configuration Register 6 (PRAMC_PRCR6)	32	R/W	0000_0100h	37.2.6/1348
18	Platform RAM Configuration Register 7 (PRAMC_PRCR7)	32	R/W	0000_0100h	37.2.7/1350
1C	Platform RAM Configuration Register 8 (PRAMC_PRCR8)	32	R/W	0000_0200h	37.2.8/1351

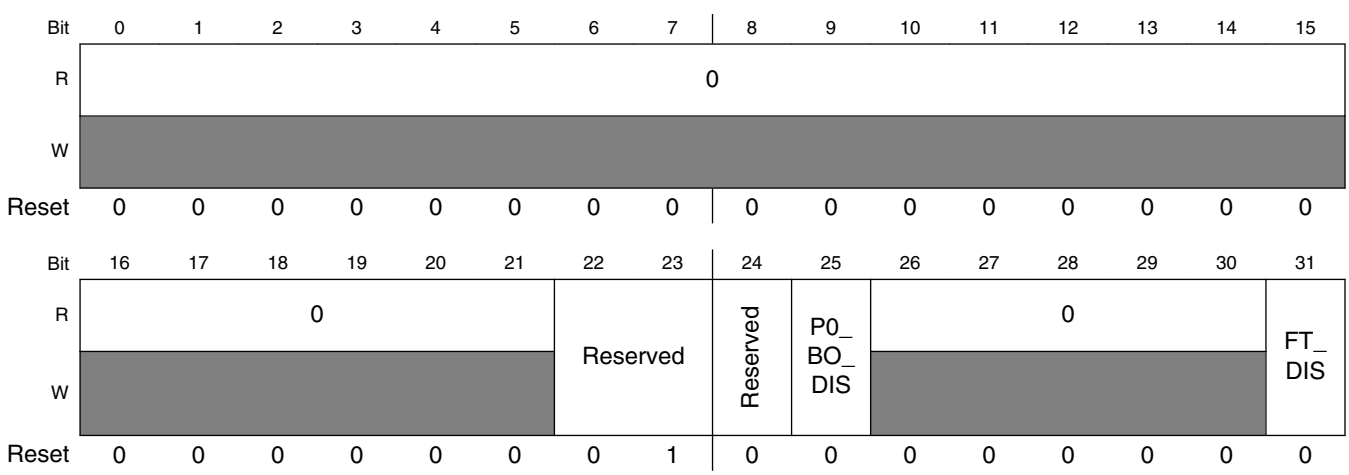
37.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 0.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h



PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

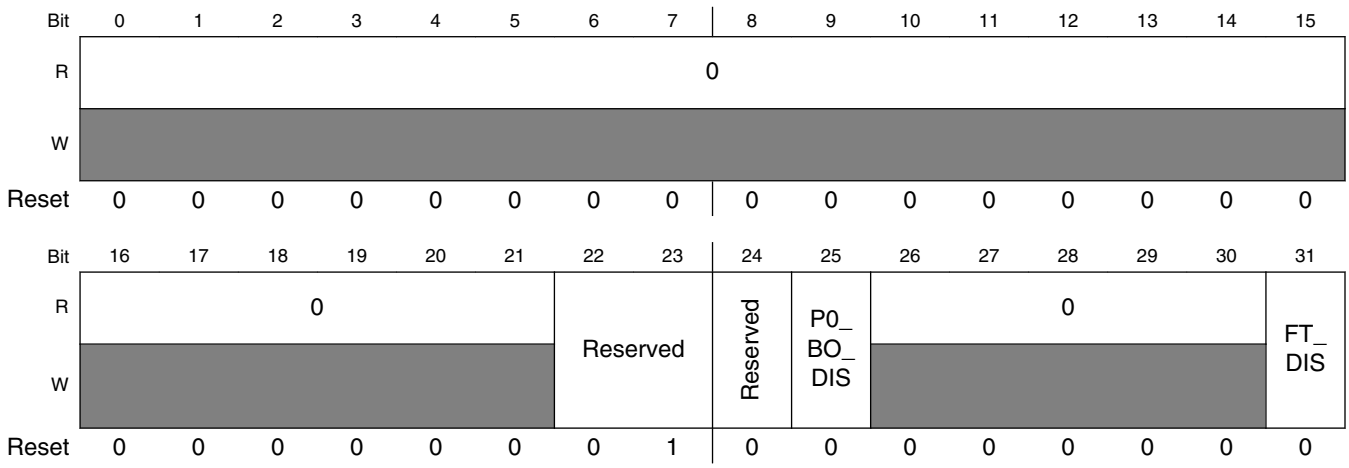
37.2.2 Platform RAM Configuration Register 2 (PRAMC_PRCR2)

The Platform RAM Configuration Register 2 (PRCR2) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 1.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 4h offset = 4h



PRAMC_PRCR2 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR2[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR2[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

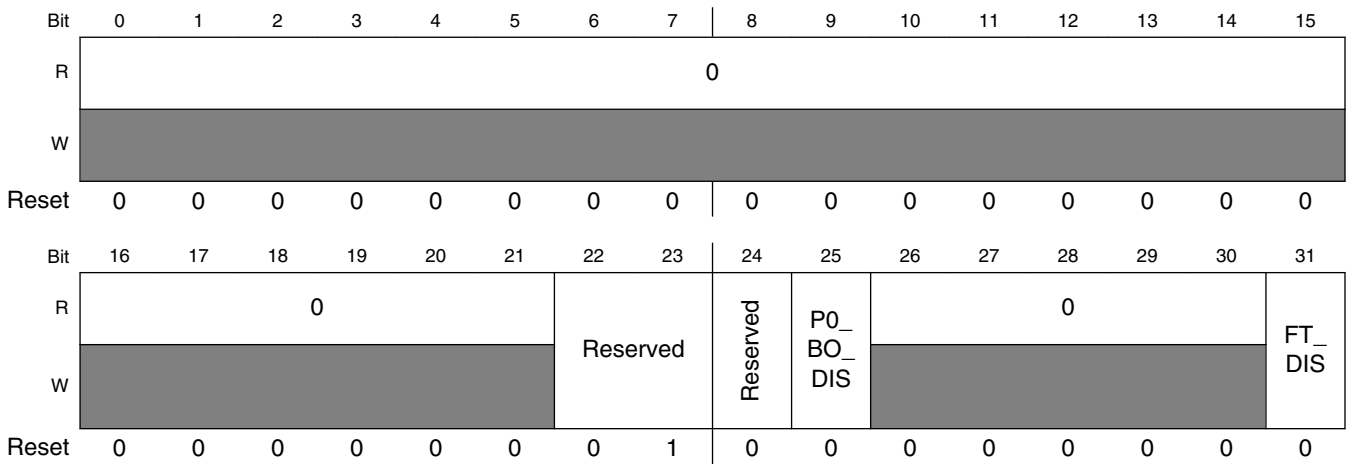
37.2.3 Platform RAM Configuration Register 3 (PRAMC_PRCR3)

The Platform RAM Configuration Register 3 (PRCR3) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 2.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 8h offset = 8h



PRAMC_PRCR3 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR3[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR3[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PRAMC_PRCR3 field descriptions (continued)

Field	Description
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

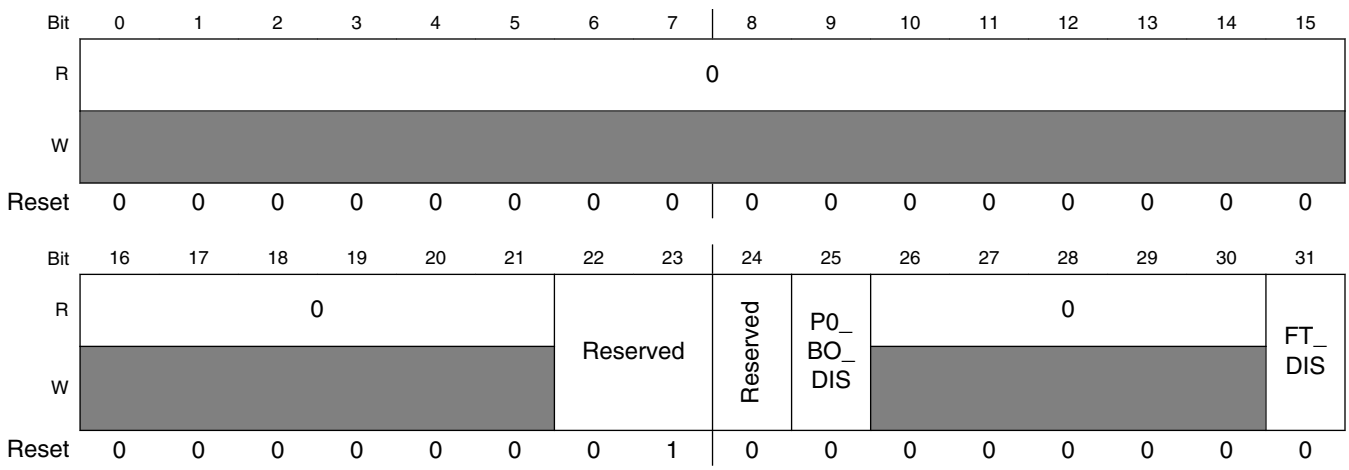
37.2.4 Platform RAM Configuration Register 4 (PRAMC_PRCR4)

The Platform RAM Configuration Register 4 (PRCR4) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 3.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + Ch offset = Ch



PRAMC_PRCR4 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.

Table continues on the next page...

PRAMC_PRCR4 field descriptions (continued)

Field	Description
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR4[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR4[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

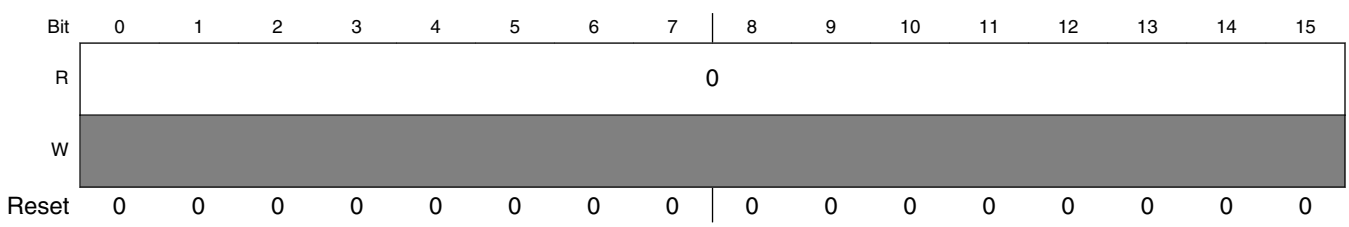
37.2.5 Platform RAM Configuration Register 5 (PRAMC_PRCR5)

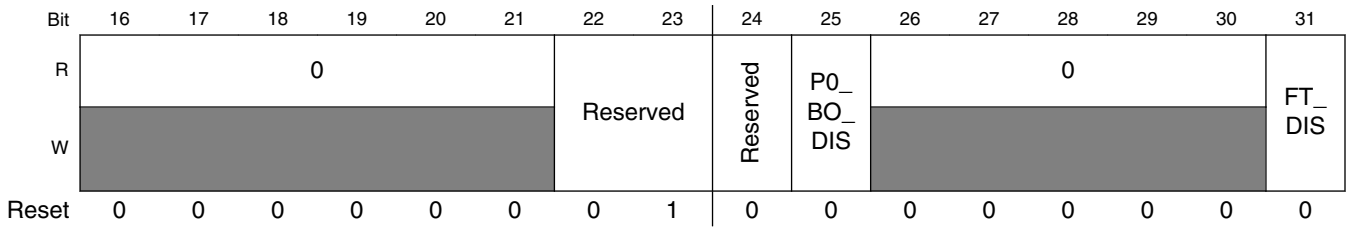
The Platform RAM Configuration Register 5 (PRCR5) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 4.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 10h offset = 10h





PRAMC_PRCR5 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR5[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR5[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

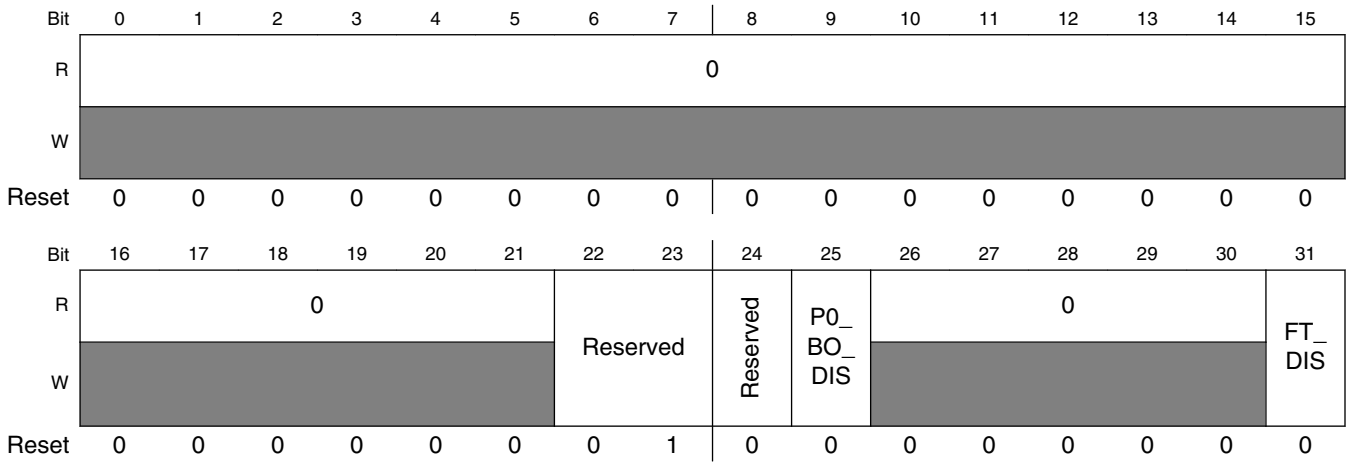
37.2.6 Platform RAM Configuration Register 6 (PRAMC_PRCR6)

The Platform RAM Configuration Register 6 (PRCR6) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 5.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 14h offset = 14h



PRAMC_PRCR6 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR6[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR6[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory.

Table continues on the next page...

PRAMC_PRCR6 field descriptions (continued)

Field	Description
	0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

37.2.7 Platform RAM Configuration Register 7 (PRAMC_PRCR7)

The Platform RAM Configuration Register 7 (PRCR7) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 6.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		Reserved	P0_BO_DIS	0					FT_DIS
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PRAMC_PRCR7 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency.

Table continues on the next page...

PRAMC_PRCR7 field descriptions (continued)

Field	Description
	0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR7[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR7[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

37.2.8 Platform RAM Configuration Register 8 (PRAMC_PRCR8)

The Platform RAM Configuration Register 8 (PRCR8) is used to specify operation of the quad-port RAM controller's internal controller for SRAM bank 7.

The internal RAM controller associated with this register is dual-ported to enable use of SRAM bank 7 as overlay RAM during system development.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PRI		P1_BO_DIS	P0_BO_DIS	0					FT_DIS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

PRAMC_PRCR8 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 PRI	AHB port arbitration mode. Use this register to select the AHB port arbitration mode. 00 Round Robin arbitration is selected. 01 Port p0 has priority over port p1. 10 Port p1 has priority over port p0. 11 Reserved.
24 P1_BO_DIS	Port p1 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR8[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR8[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR8[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR8[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

37.3 Functional description

This section describes the functions of the RAM controller.

37.3.1 Read and Write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

NOTE

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

37.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

37.3.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles

depending on RAM speed relative to the PRAMC controller clock frequency.

37.3.1.3 Writes

This section discusses various write operations of the RAM controller.

37.3.1.3.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

37.3.1.3.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. Figure 37-2 provides details on the read-modify-write datapath.

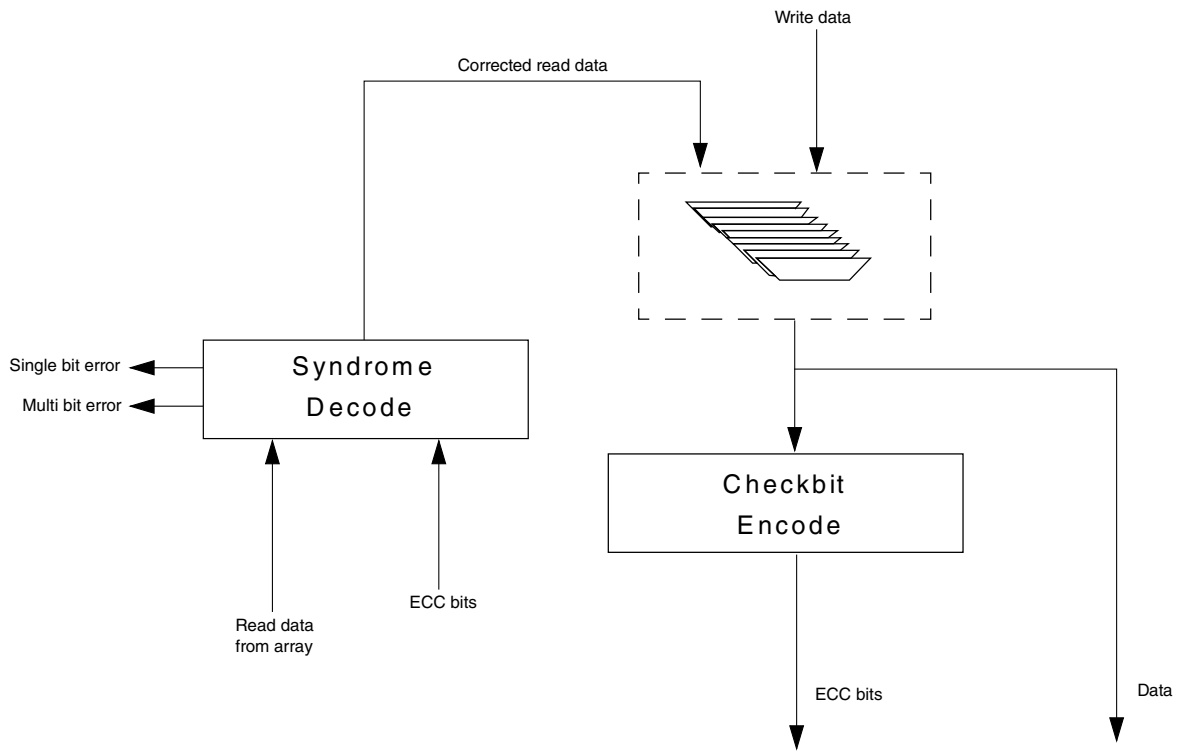


Figure 37-2. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

37.3.1.3.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

37.4 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

37.5 Reliability considerations

This section discusses reliability considerations of the RAM controller.

37.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 37-1. RAM controller basic H-matrix definition

		XOR sum	Data Bit[1]																																			
			Byte 7								Byte 6								Byte 5								Byte 4							
			63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Checkbits [7:0]	7		*				*					*			*		*	*		*	*	*	*				*	*				*	*	
	6				*	*			*	*					*		*		*				*				*			*	*			*
	5		*	*	*	*					*				*	*		*		*	*				*	*	*				*			
	4		*		*			*		*					*		*		*			*			*	*						*		
	3										*	*	*	*	*		*				*		*				*	*	*	*					
	2						*		*					*		*			*	*				*	*							*		*
	1			*		*		*		*	*	*	*	*						*		*		*		*						*		*
	0			*			*	*	*				*			*	*	*		*				*				*		*	*		*		
		XOR sum	Byte 3								Byte 2								Byte 1								Byte 0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7						*		*				*		*				*	*			*	*			*				*	*		*
	6		*	*		*	*				*				*				*	*		*	*					*	*				*	*
	5				*	*			*	*	*			*	*	*					*		*	*		*								
	4		*		*	*				*		*	*				*	*	*			*			*	*				*		*		
	3		*	*	*			*						*	*		*	*			*				*		*	*	*			*		
	2			*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
	1			*						*		*			*	*	*			*				*				*			*	*	*		
	0			*		*		*		*				*				*				*				*					*	*		*	*
		XOR sum	Address Bit[2]																																				
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7			*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*		Unused	Unused	Unused
	6			*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
	5			*	*	*			*	*			*	*		*		*		*	*			*	*			*	*	*	*				
	4		*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*					
	3		*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*				
	2		*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*				
	1		*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*				
	0		*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*				

1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

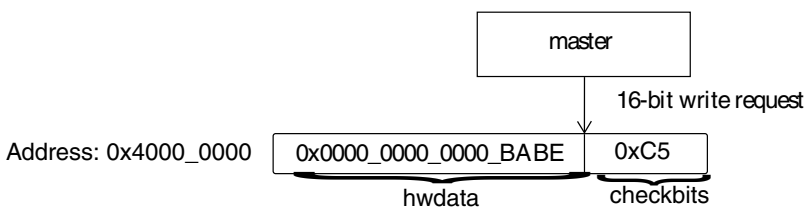
2. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

37.5.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

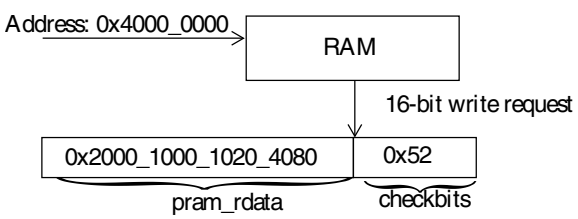
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

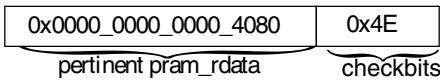


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

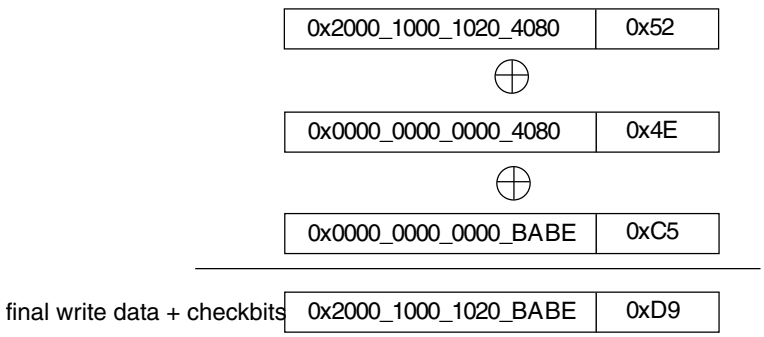
  If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting.

37.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions:

- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor also verifies the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

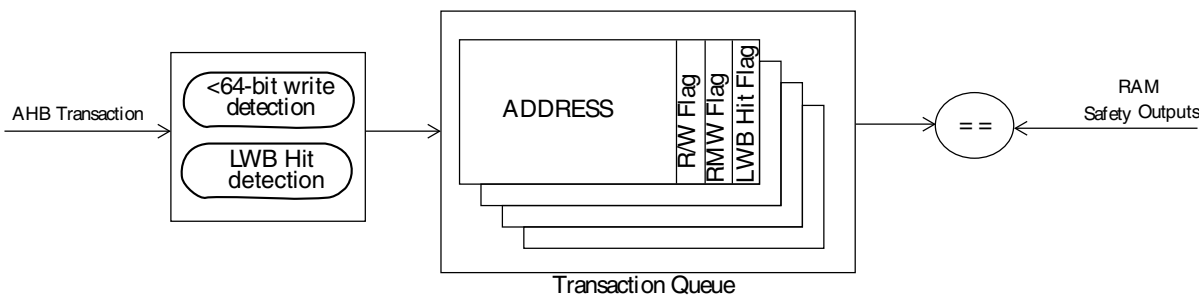


Figure 37-3. RAM controller transaction monitor block diagram

Chapter 38
Flash memory controller

38.1 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to on-chip System RAM that can be used as overlay RAM.

The flash memory controller supports three 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 256 bytes of total buffered storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

38.2 Features

The following list summarizes the key features of the flash memory controller:

- Three 64-bit AHB interface (p0, p1, p2) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.

- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip system RAM

38.3 Block diagrams

Figure 38-1 illustrates the connections of the flash memory controller and flash memory array within the chip.

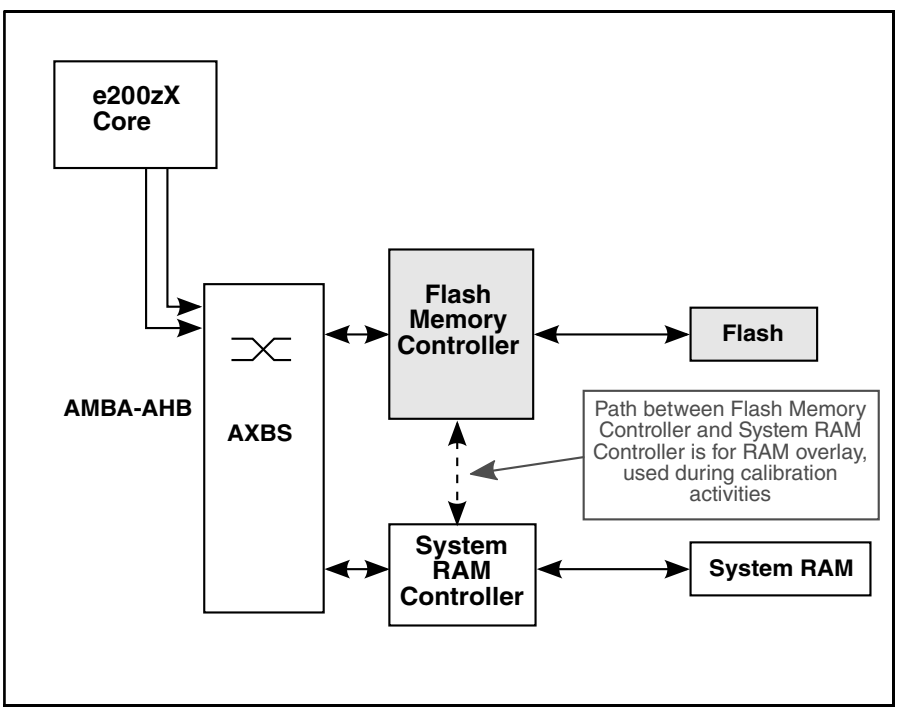


Figure 38-1. Platform-centric simple block diagram with flash memory controller

38.4 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configuration and overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

A write to any flash memory controller register, when executed concurrently with a flash memory access, will result in non-deterministic behavior and may include lockup of the flash memory controller. There is no idle indicator for the flash controller, so software must be architected to prevent any possible conflicts. The following recommendations will help.

- For multi-core devices, start only one core and execute initialization code to completion before starting the remaining cores.
- If flash must be reconfigured during application code execution, temporarily move all required code and data to system RAM before any writes to flash memory controller registers.
- If application software functionality includes invalidating a port read buffer, the recommended method for invalidating a port read buffer is:
  a. Unlock the buffer's enable field by setting the PFLASH_PFCR3[BFEN_LK] field to '0'
  b. Disable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '0'
  c. If needed, enable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '1'

The above method requires updating flash memory controller registers, so software must ensure that no flash accesses are issued concurrently with any of the steps. In cases where that cannot be guaranteed, there is a workaround but its use is not generally recommended because it relies on a hardware side effect instead of the flash memory controller's built-in functions intended for invalidating port read buffers. The workaround is to issue an interlock write without an accompanying program or erase operation—an interlock write, without an accompanying program or erase, does not change any flash content.

NOTE

1. If this workaround is implemented in application code, note that an interlock write invalidates only the buffers associated with the system bus port on which the interlock write was issued.
2. See the flash memory chapter for the steps to issue an interlock write.

3. See the flash memory controller's chip-specific information for any restrictions on the interlock write data size.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	38.4.1/1365
4	Platform Flash Configuration Register 2 (PFLASH_PFCR2)	32	R/W	0000_0001h	38.4.2/1368
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	38.4.3/1371
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	38.4.4/1373
10	Platform Flash Remap Control Register (PFLASH_PFCRCR)	32	R/W	0000_0000h	38.4.5/1376
14	Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)	32	R/W	0000_0000h	38.4.6/1378
18	Platform Flash Configuration Register 4 (PFLASH_PFCR4)	32	R/W	0000_0001h	38.4.7/1380
100	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD0_Word0)	32	R/W	0000_0000h	38.4.8/1383
104	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD0_Word1)	32	R/W	0000_0000h	38.4.9/1383
108	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD0_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
110	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD1_Word0)	32	R/W	0000_0000h	38.4.8/1383
114	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD1_Word1)	32	R/W	0000_0000h	38.4.9/1383
118	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD1_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
120	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD2_Word0)	32	R/W	0000_0000h	38.4.8/1383
124	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD2_Word1)	32	R/W	0000_0000h	38.4.9/1383
128	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD2_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
130	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD3_Word0)	32	R/W	0000_0000h	38.4.8/1383
134	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD3_Word1)	32	R/W	0000_0000h	38.4.9/1383
138	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD3_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
140	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD4_Word0)	32	R/W	0000_0000h	38.4.8/1383
144	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD4_Word1)	32	R/W	0000_0000h	38.4.9/1383
148	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD4_Word2)	32	R/W	0000_0000h	38.4.10/ 1385

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
150	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD5_Word0)	32	R/W	0000_0000h	38.4.8/1383
154	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD5_Word1)	32	R/W	0000_0000h	38.4.9/1383
158	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD5_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
160	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD6_Word0)	32	R/W	0000_0000h	38.4.8/1383
164	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD6_Word1)	32	R/W	0000_0000h	38.4.9/1383
168	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD6_Word2)	32	R/W	0000_0000h	38.4.10/ 1385
170	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD7_Word0)	32	R/W	0000_0000h	38.4.8/1383
174	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD7_Word1)	32	R/W	0000_0000h	38.4.9/1383
178	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD7_Word2)	32	R/W	0000_0000h	38.4.10/ 1385

38.4.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_ BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. A value of '1' in the most significant bit causes the flash controller to function in retrograde/ legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. Higher operating frequencies require non-zero settings for this field for proper flash operation. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet. This field applies to the configuration of Port0, Port1, and Port2. 00000 No additional wait-states are added

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

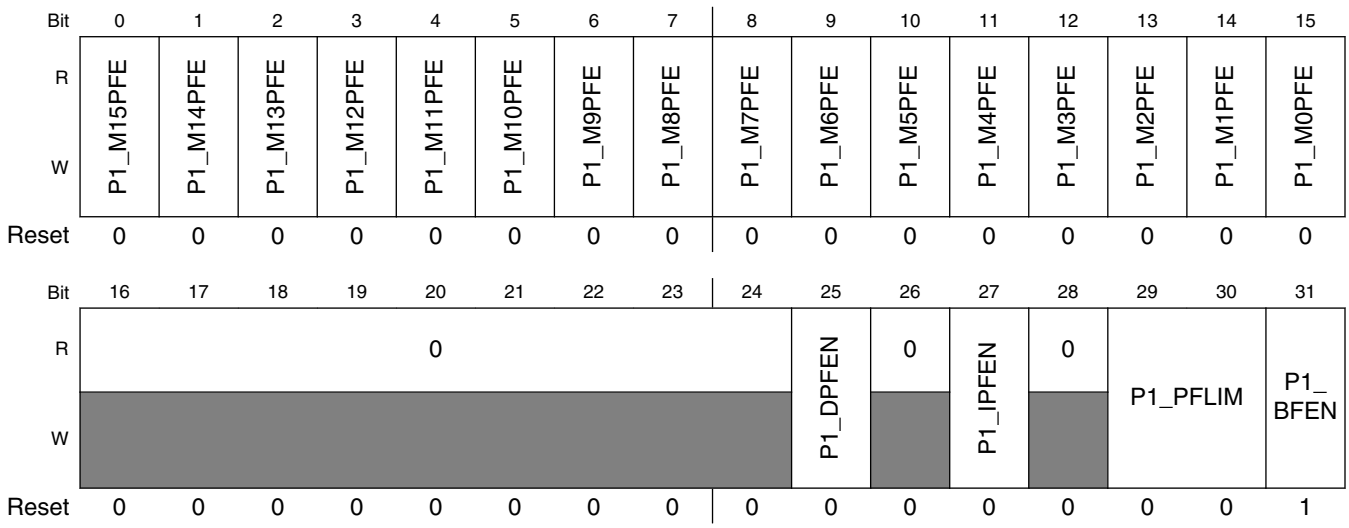
38.4.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)

The PFlash Configuration Register 2 (PFCR2) controls the operation of Port1 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for details about the masters.

Address: 0h base + 4h offset = 4h



PFLASH_PFCR2 field descriptions

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access.

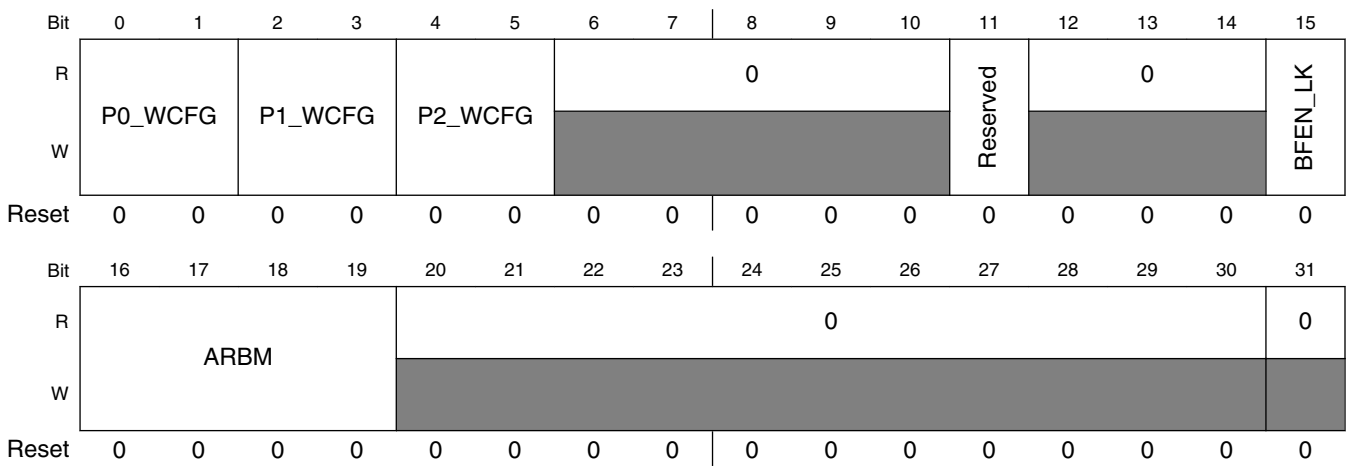
Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

38.4.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h



PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
2–3 P1_WCFG	Port1 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
4–5 P2_WCFG	Port2 Line Buffer Configuration. This field controls the configuration of the line buffers for a given set across both ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffer 0 allocated for instruction fetches and buffer 1 for data accesses. 11 Reserved
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
15 BFEN_LK	BFEN Lock. This field controls access to the PFCRm[Pn_BFEN] fields, which enable or disable line read buffer hits and are also used to invalidate the buffers. After this field is set to 1, it can be cleared to 0 only by a system reset. 0 PFCRm[Pn_BFEN] fields are R/W. 1 PFCRm[Pn_BFEN] fields are Read-only.
16–19 ARBM	Arbitration Mode. This 4-bit field controls the arbitration of concurrent flash access requests from the three AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever multiple ports issue concurrent requests. This field is initialized by hardware reset. **NOTE:** Specifying a reserved value in this field causes it to default to round-robin arbitration. 0000 Fixed priority arbitration with AHB p0>p1>p2 0001 Fixed priority arbitration with AHB p0>p2>p1 0010 Fixed priority arbitration with AHB p2>p0>p1 0011 Fixed priority arbitration with AHB p2>p1>p0 0100 Fixed priority arbitration with AHB p1>p0>p2 0101 Fixed priority arbitration with AHB p1>p2>p0 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Round-robin arbitration
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

38.4.4 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
	01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
	01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

38.4.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

CAUTION

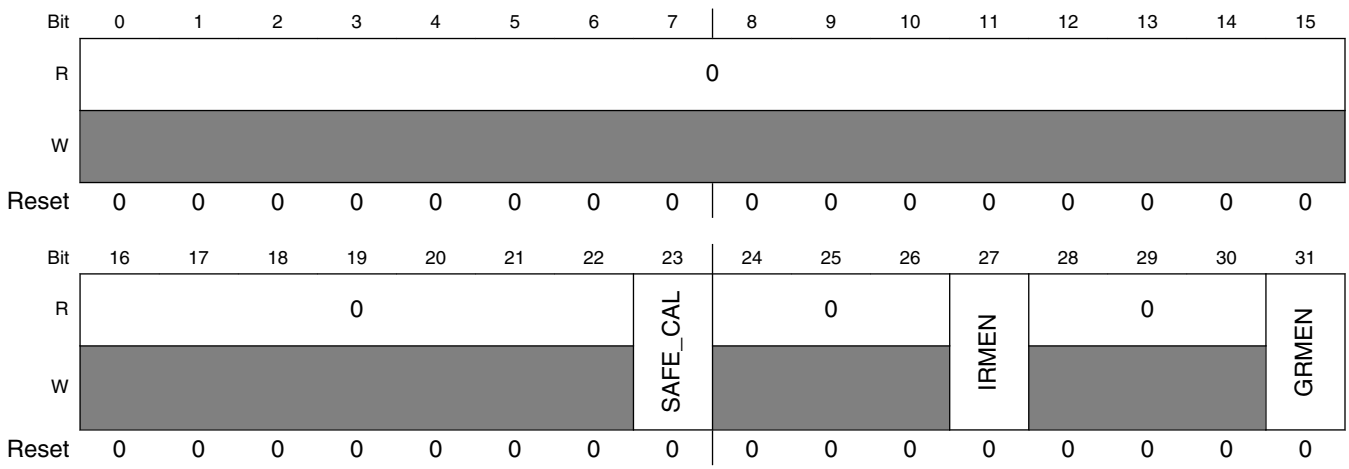
Overlay remap functions, when used with line read buffers enabled (PFLASH_PFCRx[BFy_EN] field(s)='1') have a potential to cause data coherency issues, i.e., prefetched data in a line read buffer can become out-of-sync with data contained at the associated flash address range. To prevent this issue, always clear the flash controller line read buffers prior to enabling any remap region.

To clear the flash controller line read buffers, issue an interlock write command to a valid flash memory address. There will not be an actual flash program operation performed, but the flash controller mini-cache will be cleared as a side effect of

executing the command. The interlock write must be issued from a core having appropriate access to flash. The sequence is as follows:

1. Set the C55FMC_MCR[PGM] field to '1'
2. Issue a 64-bit write to a valid flash address on a 64-bit boundary, i.e., the least significant 5 bits of the address are '0'
3. Clear C55FMC_MCR[PGM]

Address: 0h base + 10h offset = 10h



PFLASH_PFCRCR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 8 to 4. This field is initialized by hardware reset. 1 Established calibration overlay regions are considered safety-critical. 0 Established calibration overlay regions are not considered safety-critical.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PFLASH_PFCRCR field descriptions (continued)

Field	Description
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

38.4.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 8 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 0 invalid 1 Calibration Remap Descriptor 0 valid

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 1 invalid 1 Calibration Remap Descriptor 1 valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 2 invalid 1 Calibration Remap Descriptor 2 valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 3 invalid 1 Calibration Remap Descriptor 3 valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 4 invalid 1 Calibration Remap Descriptor 4 valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 5 invalid 1 Calibration Remap Descriptor 5 valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 6 invalid 1 Calibration Remap Descriptor 6 valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 7 invalid 1 Calibration Remap Descriptor 7 valid
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

38.4.7 Platform Flash Configuration Register 4 (PFLASH_PFCR4)

The PFlash Configuration Register 4 (PFCR4) specifies operation of Port2 of the flash memory controller.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P2_M15PFE	P2_M14PFE	P2_M13PFE	P2_M12PFE	P2_M11PFE	P2_M10PFE	P2_M9PFE	P2_M8PFE	P2_M7PFE	P2_M6PFE	P2_M5PFE	P2_M4PFE	P2_M3PFE	P2_M2PFE	P2_M1PFE	P2_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									P2_DPFEN	0	P2_IPFEN	0	P2_PFLIM		P2_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PFLASH_PFCR4 field descriptions

Field	Description
0 P2_M15PFE	Port2 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P2_M14PFE	Port2 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR4 field descriptions (continued)

Field	Description
2 P2_M13PFE	Port2 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P2_M12PFE	Port2 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P2_M11PFE	Port2 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P2_M10PFE	Port2 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P2_M9PFE	Port2 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P2_M8PFE	Port2 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P2_M7PFE	Port2 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P2_M6PFE	Port2 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P2_M5PFE	Port2 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P2_M4PFE	Port2 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P2_M3PFE	Port2 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P2_M2PFE	Port2 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2.

Table continues on the next page...

PFLASH_PFCR4 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P2_M1PFE	Port2 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P2_M0PFE	Port2 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P2_DPFEN	Port2 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P2_IPFEN	Port2 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P2_PFLIM	Port2 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P2_BFEN	PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

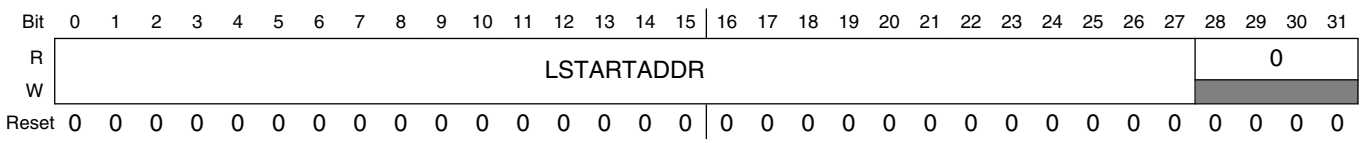
38.4.8 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 100h offset + (16d × i), where i=0d to 7d



PFLASH_PFCRD*n*_Word0 field descriptions

Field	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

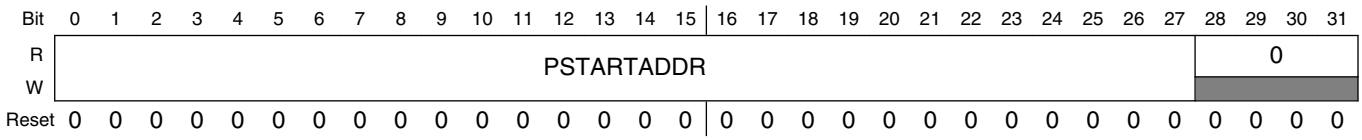
38.4.9 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region.

The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 104h offset + (16d × i), where i=0d to 7d



PFLASH_PFCRD*n*_Word1 field descriptions

Field	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

38.4.10 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Address: 0h base + 108h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word2 field descriptions

Field	Description
0 M0EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 0 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 0 as defined by this region descriptor.
1 M1EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 1 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 1 as defined by this region descriptor.
2 M2EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 2 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 2 as defined by this region descriptor.
3 M3EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 3 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 3 as defined by this region descriptor.
4 M4EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 4 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 4 as defined by this region descriptor.
5 M5EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 5 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 5 as defined by this region descriptor.
6 M6EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 6 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 6 as defined by this region descriptor.
7 M7EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 7 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 7 as defined by this region descriptor.
8 M8EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 8 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 8 as defined by this region descriptor.
9 M9EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 9 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 9 as defined by this region descriptor.
10 M10EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 10 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 10 as defined by this region descriptor.
11 M11EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 11 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 11 as defined by this region descriptor.
12 M12EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 12 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 12 as defined by this region descriptor.
13 M13EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 13 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 13 as defined by this region descriptor.
14 M14EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 14 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 14 as defined by this region descriptor.
15 M15EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 15 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 15 as defined by this region descriptor.
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved 00100 Reserved 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero
	01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero
	01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero
	01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero
	10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero
	10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero
	10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero
	10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero
	10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero
	10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero
	10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero
	10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero
	11000 Reserved
	11001 Reserved
	11010 Reserved
	11011 Reserved
	11100 Reserved
	11101 Reserved
	11110 Reserved
	11111 Reserved

38.5 Functional description

The flash memory controller interfaces between:

- The AHB system bus ports
- Flash memory array
- System RAM

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

38.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

38.5.2 Access protections

On-chip Flash memory accesses are subject to restrictions based on chip security implementation and/or requirements based on functional restrictions of the flash memory, e.g., a read access to a flash block while it is currently being written is not allowed. When a prohibited access is attempted, a system bus error termination results.

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding read access control or write access control settings do not allow the access, thus causing a protection violation. See Platform Flash Access Protection Register (PFLASH_PFAPR) for more detail. In this case the flash memory controller does not initiate the requested flash array access.
- The flash returns an error response on an attempted access to a partition that is unavailable. (see Flash error response operation or Embedded Flash Memory).
- Attempted access by an AHB master to a reserved region in the flash memory map.

The flash array may also signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

38.5.2.1 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Platform Flash Access Protection Register (PFLASH_PFAPR). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

38.5.3 Security Module Alternate Programming Interface

The secure flash space is independently controlled for each of the following regions:

- 16 KB CSE2 flash block0 at address range 00A0_0000h - 00A0_3FFFh
- 16 KB CSE2 flash block1 at address range 00A0_4000h - 00A0_7FFFh

38.5.4 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

NOTE

Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

38.5.5 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Platform Flash Configuration Register 1 (PFLASH_PFCR1)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

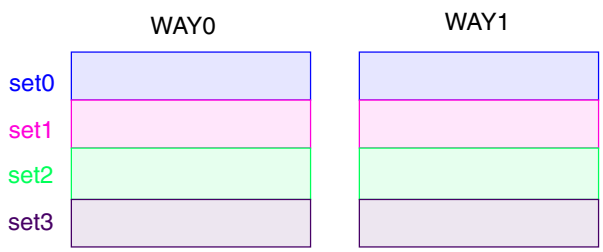


Figure 38-2. Flash memory controller 4-entry, 2-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Platform Flash Configuration Register 1 (PFLASH_PFCR1). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2,4}[P{0,1,2}_BFEN] must be set to ‘1’,

2. PFCR{1,2,4}[P{0,1,2}_PFLIM] must be non-zero, and
3. Either PFCR{1,2,4}[P{0,1,2}_IPFEN] or PFCR{1,2,4}[P{0,1,2}_DPFEN] must be asserted.

Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

38.5.6 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to Platform Flash Configuration Register 2 (PFLASH_PFCR2) for a description of these controls. Prefetches are never triggered by write cycles.

Port2 prefetch triggering may be enabled for instruction reads via the PFCR4[IPFEN] control field, while Port2 prefetching for data reads is enabled via the PFCR4[DPFEN] control field. Additionally, the PFCR4[PFLIM] must also be set to enable prefetching on Port2. Refer to the Platform Flash Configuration Register 4 (PFCR4) section for a description of these controls. Prefetches are never triggered by write cycles.

38.5.7 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

38.5.8 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1,P2}_WCFG) field. Refer to Platform Flash Configuration Register 3 (PFLASH_PFCR3). The buffers can be organized as a "pool" of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

38.5.9 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to on-chip System RAM that can be used as overlay RAM.

The overlay function supports the following features:

- Can be mapped over internal flash memory
  - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash
  - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function :
  - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
  - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay targets:

- A portion of the system RAM can be used for overlay.

38.5.9.1 PFlash calibration remap to RAM

The overlay RAM is selected based on the translated physical address.

The translated physical address defines the destination memory. Table 38-1 shows the base addresses for the calibration data memories as defined by the system memory map.

Table 38-1. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address.

The read data interface of system RAM is 64 bits wide , whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(\mathrm{RWSC}+2) \geq \left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right) \geq (\mathrm{OverlayRAMTransferBurstLatency})$$

Equation 12. Access time

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150 °C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries.

The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap

descriptor to determine the region "hit". The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region "hit" data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

38.5.10 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

38.5.10.1 e2eECC End-to-End ECC

38.5.10.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC

implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

   addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

   addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

   Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

   fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller

inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

38.5.10.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

38.5.10.2.1 Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within

the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

38.5.10.2.2 Reliability considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the flash memory controller treats the set of available calibration region descriptors as a replicated pair, with each set containing half the total number of available calibration region descriptors.

If there are 32 calibration region descriptors, CRD0 – CRD31, and PFCRCR[SAFE_CAL]=1, the flash memory controller treats CRD0 – CRD15 and CRD16 – 32 as replicated sets of 16 calibration region descriptors, where:

- CRD0 and CRD16 are configured identically
- CRD1 and CRD17 are configured identically
- CRD2 and CRD18 are configured identically
- ...
- and CRD15 and CRD31 are configured identically

Expressed more generally, the contents of CRD[n] must be identical to the contents of CRD[(N/2) + n], where N is the total number of region descriptors and $0 \le n \le (N/2) - 1$.

Note

It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2) – 1] and the calibration regions defined in CRD[N/2] to CRD[N – 1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

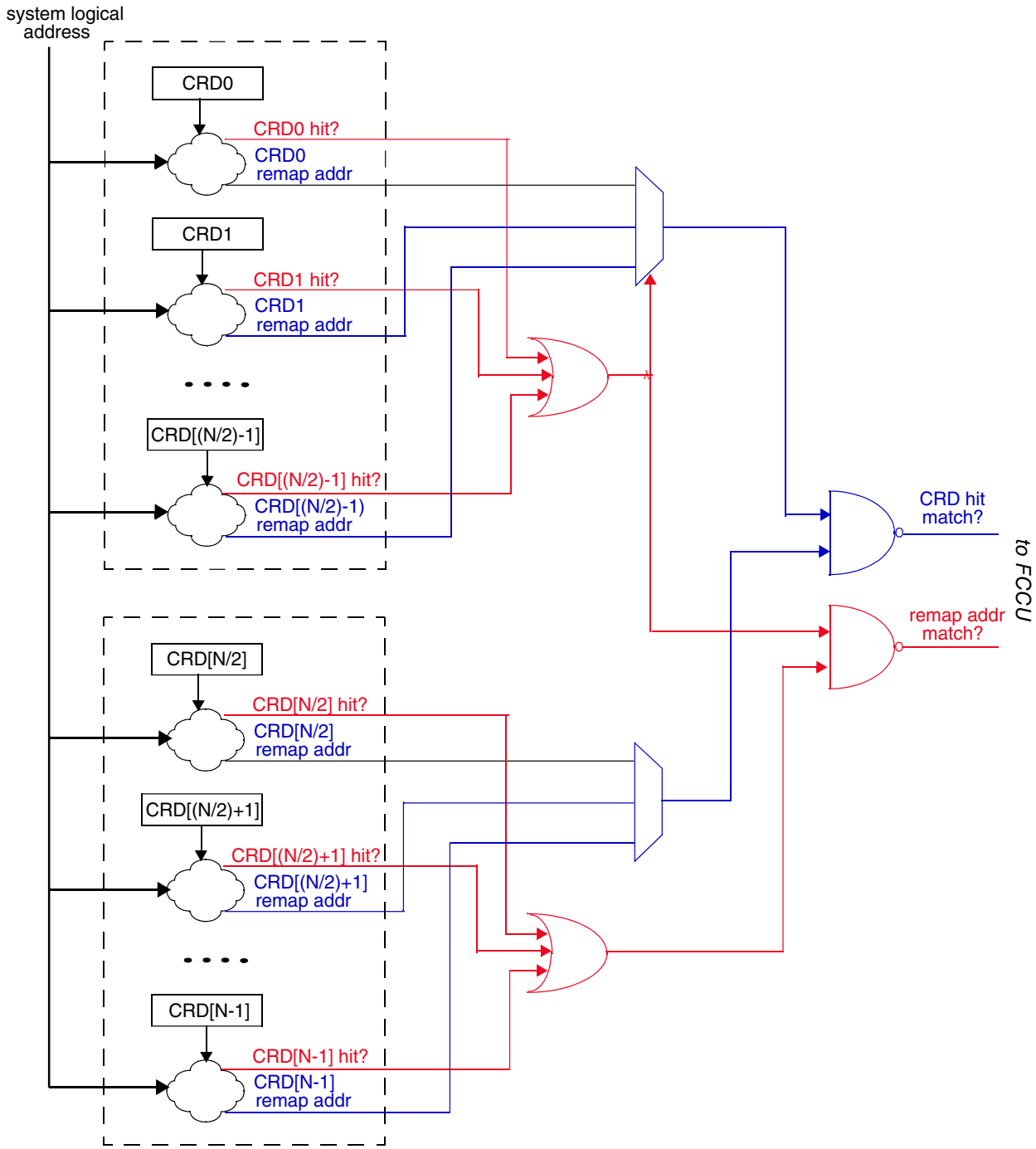


Figure 38-3. Safety-critical calibration remap datapath with redundancy

38.5.11 ECC on data flash accesses

For single-bit correction events, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value (1555_1555h) is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address and the non-correctable error event is reported to the MEMU.

NOTE

EEPROM should be avoided for storage of executable code.

38.5.12 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 39
Embedded Flash Memory (c55fmc)

39.1 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

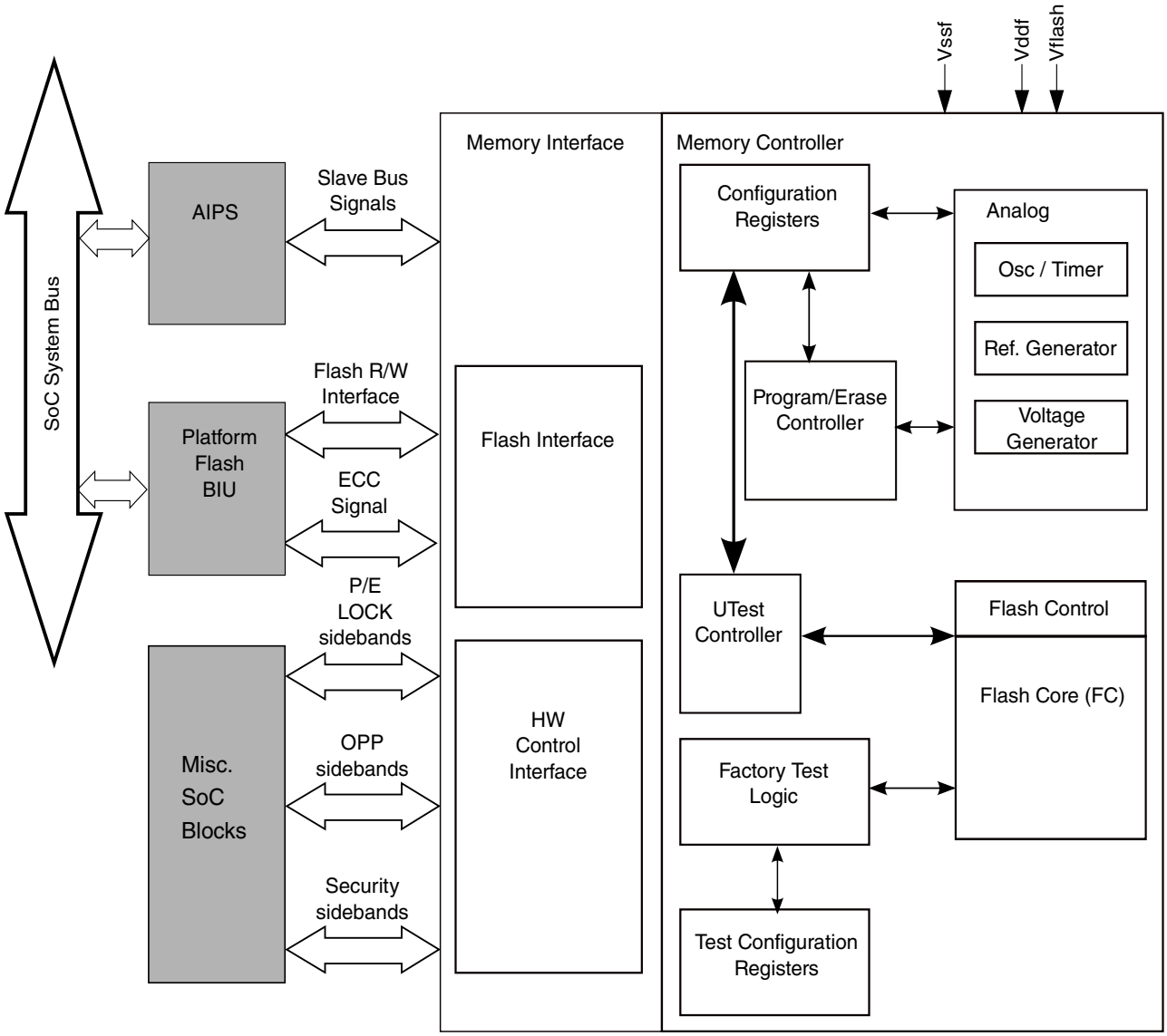


Figure 39-1. c55fmc block diagram

39.1.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see Functional Description.

39.1.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

39.1.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

39.2 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

39.3 Test mode disable seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the UTest block becomes OTP (Erase Locked, and Over Program Protection enabled). The UTest Block also is not accessible in Test mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

UTest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and single bit corrections and double bit detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A. Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written. UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field(s) are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to provide two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 39-1. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

39.4 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks.

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	39.4.1/1409
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	39.4.2/1414
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	39.4.3/1417
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	39.4.4/1419
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	39.4.5/1419
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	39.4.6/1420
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	39.4.7/1421
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	39.4.8/1422
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	39.4.9/1423
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	39.4.10/ 1424
50	Address register (C55FMC_ADR)	32	R	0000_0000h	39.4.11/ 1425
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	39.4.12/ 1427
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	39.4.13/ 1430
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	39.4.13/ 1430
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	39.4.13/ 1430
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	39.4.13/ 1430
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	39.4.13/ 1430
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	39.4.13/ 1430
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	39.4.13/ 1430
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	39.4.13/ 1430
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	39.4.13/ 1430
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	39.4.14/ 1431
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	39.4.15/ 1432
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	39.4.16/ 1433
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	39.4.17/ 1434
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	39.4.18/ 1434
90	Test Mode Disable Password Check register (C55FMC_TMD)	32	R/W	0000_0000h	39.4.19/ 1435

39.4.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description.
2. See Functional Description for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	0			PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations(and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. **NOTE:** This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. **NOTE:** 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. **NOTE:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

39.4.2 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0		n256K					n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. **NOTE:** 165 °C operating temperature maximum is not supported on this device. 1 Reserved
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total).

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	00101 Ten 256 KB blocks (2.5 MB total). 00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24–26 n64Kl	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32Kl	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table continues on the next page...

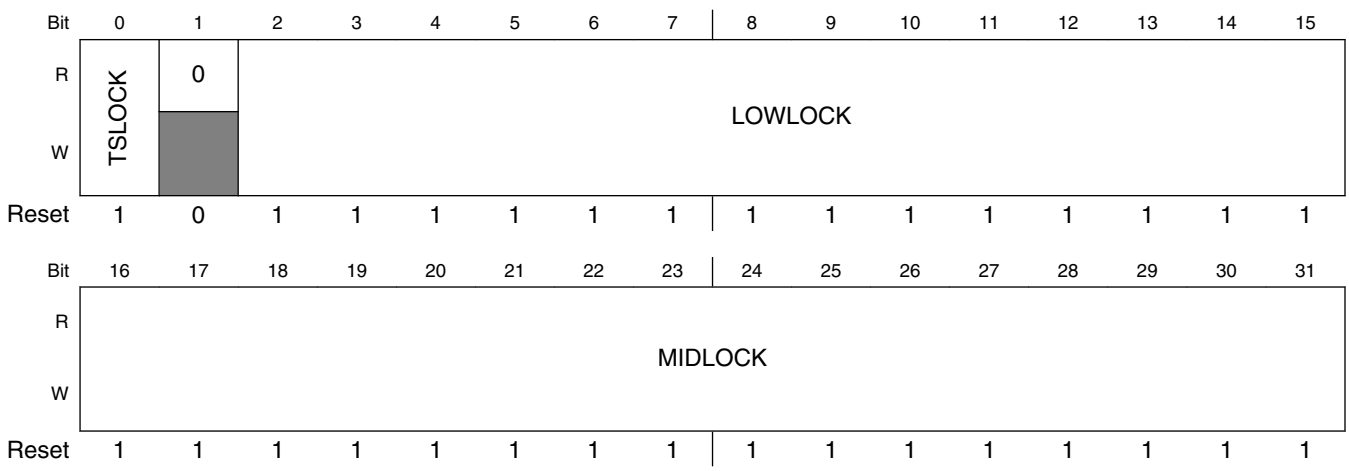
C55FMC_MCRE field descriptions (continued)

Field	Description
29–31 n16Kl	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

39.4.3 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h



C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended.

Table continues on the next page...

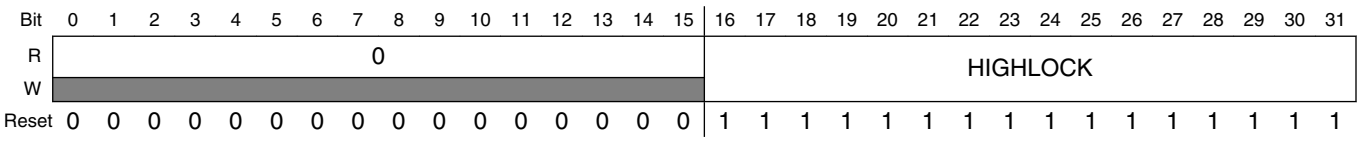
C55FMC_LOCK0 field descriptions (continued)

Field	Description
	0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

39.4.4 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h



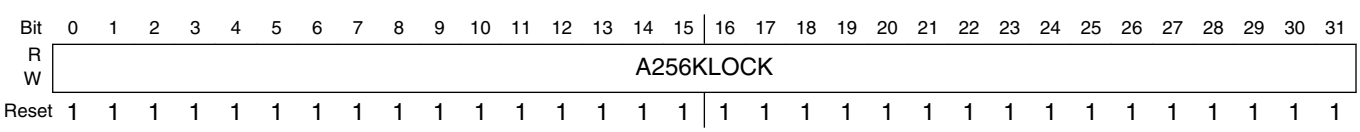
C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

39.4.5 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h



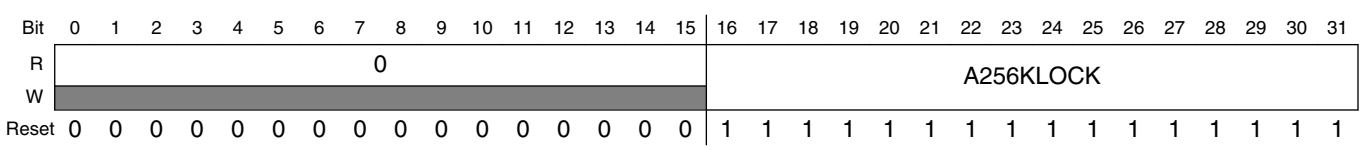
C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

39.4.6 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch



C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted.

Table continues on the next page...

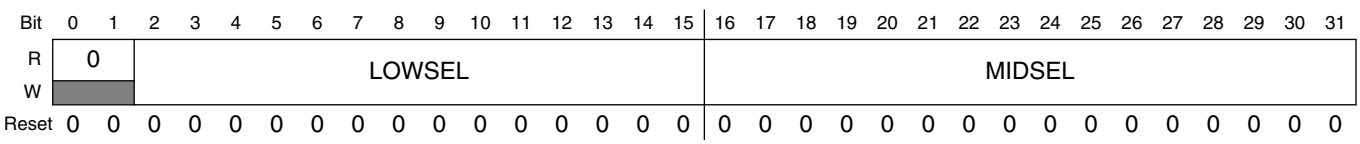
C55FMC_LOCK3 field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

39.4.7 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h



C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

39.4.8 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

39.4.9 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

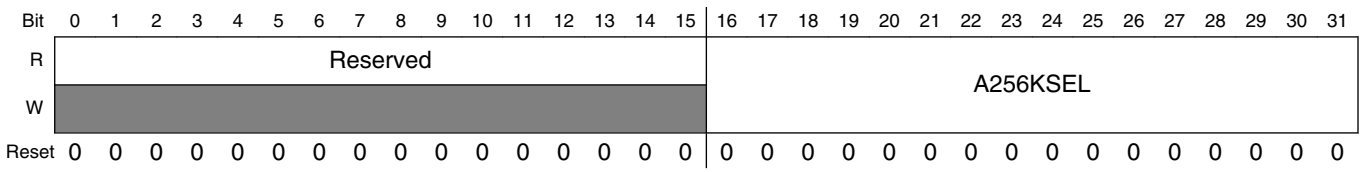
C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

39.4.10 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h



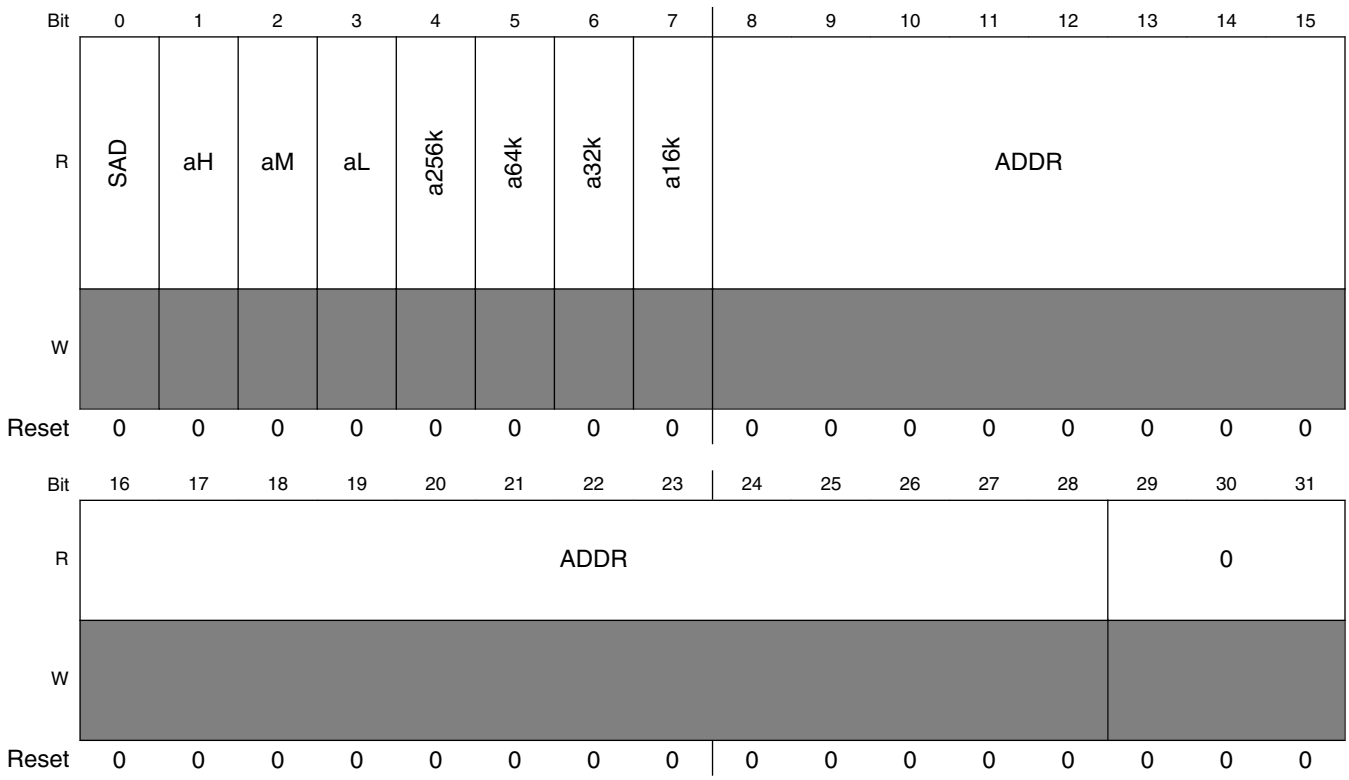
C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

39.4.11 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h



C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

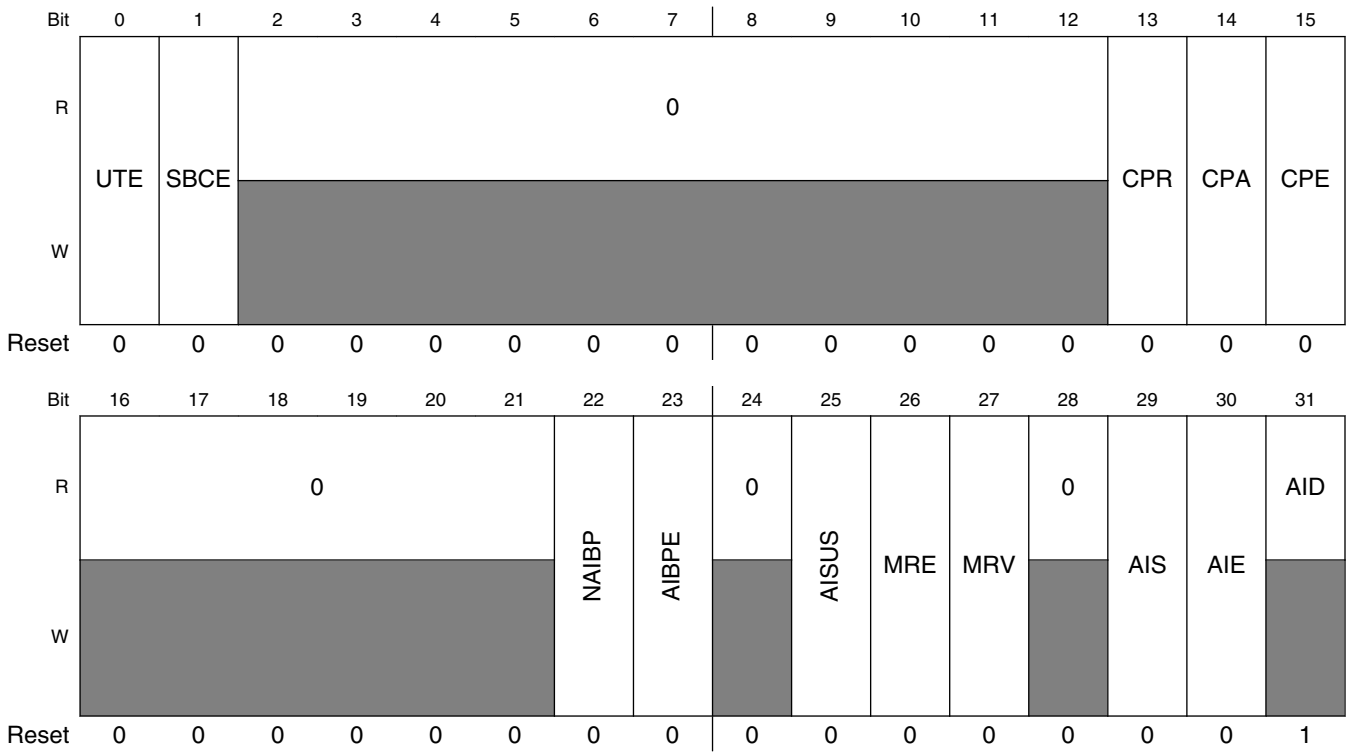
C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

39.4.12 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h



C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. **NOTE:** Only sequential addressing is supported for margin read checks. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

39.4.13 UMISR register (C55FMC_UM*n*)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM*n* field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

39.4.14 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

39.4.15 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWOPP														MIDOPP															
W																																
Reset	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

39.4.16 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

39.4.17 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 0 The corresponding block is available to be over-programmed.

39.4.18 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

39.4.19 Test Mode Disable Password Check register (C55FMC_TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_TMD field descriptions

Field	Description
0–31 PWD	Password challenge

39.5 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register (C55FMC_MCR). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 39-2. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

39.5.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

39.5.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] =0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

39.5.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

   NOTE

   Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as

a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.

3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

39.5.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

39.5.1.2.2 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend

frequency is not limited, enabling at least 20 µs between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

39.5.1.2.3 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

39.5.1.2.4 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset.

39.5.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.

2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored. The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

39.5.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see Program software locking.

39.5.1.3.2 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase

suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

NOTE

1. In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.
2. Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 5ms between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

39.5.2 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

39.5.2.1 Array integrity self check

The Array Integrity (AI) operation is a flash-specific self-test feature in which a MISR signature value is generated from data read from selected flash blocks. After the operation is complete, customer code compares the resulting MISR signature to the expected value to determine if an incorrect read or ECC detection occurred.

Array integrity operations are fully contained within the flash module. Results of Array integrity operations are not reported beyond the boundary of the flash module.

During array integrity test, user mode array reads are ignored to ensure the array integrity operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

AI read operations are executed using a predefined address sequence selected via the UT0[AIS] register field value. The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or nonrandom code is valid. The expected MISR signature is calculated based on the specific data to be read.

Note that array integrity MISR signature calculations are performed after ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

Array integrity processing requires the flash memory controller's PFLASH_PFCR1[RWSC] and PFLASH_PFCR1[APC] register fields to be set to values that are valid for the system frequency being used. Only the valid/recommended RWSC and APC values for a given frequency, as stated in the device data sheet, are supported.

The Array integrity check consists of the following sequence:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for an array integrity check by writing 1's to the appropriate bits in the SEL0 – SEL3 registers. Blocks selected for AI test do not need to be unlocked.

Flash blocks that have not been selected are still read as part of the array integrity sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

NOTE

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

   For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that UT0[AIS] be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] must be set.
6. Initiate AI by setting the UT0[AIE] bit to 1.

   The array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers contain a signature for the portion of the operation that was completed prior to the abort, and are not deterministic. Prior to doing another array integrity operation, the UM0 – UM9 registers may need to be initialized to the desired seed value by doing register writes.

   The array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the flash may be done. After the UT0[AID] bit goes high, the UT0[AISUS] bit may be cleared to resume the array integrity sequence.

7. Wait until the UT0[AID] bit goes high, indicating that array integrity test is complete.

8. If breakpoints are enabled, check the UT0[NAIBP] bit. If set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear the MCR[SBC] or MCR[EER] bits, then resume the operation by clearing the UT0[NAIBP] bit. Continue to wait until the UT0[AID] bit goes high. If breakpoints are not enabled, or if the UT0[NAIBP] bit is low when the UT0[AID] bit goes high, the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit to complete array integrity test.

39.5.2.2 User margin read

User Margin Read is a flash-specific self-test feature that enables reads to be done at a Program margin or Erase margin level.

Margin reads are performed with an internal read reference voltage that has been adjusted, i.e., "margined" toward the threshold voltage being checked. The read reference voltage is either margined toward a 0 (programmed bit) or margined towards a 1 (erased bit), depending on the margin read operation selected.

User margin read is done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin read is a self-timed event, and is independent of system clocks or wait states selected.

Margin read operations are fully contained within the flash module. Results of margin read operations are not reported beyond the boundary of the flash module.

During margin read operations, user mode array reads are ignored to ensure the operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

ECC corrections and detections are noted during the user margin read test. Note that margin read MISR signature calculations are performed *after* ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or non-random code is valid. The expected MISR signature is calculated based on the specific data to be read. After the operation is completed the margin read results can be checked by reading the MCR[EER] and MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, and comparing the generated MISR signature to the expected value.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for margin read check by writing 1's to the appropriate bits in SEL0 – SEL3 registers. Blocks selected for margin read do not need to be unlocked.

   Flash blocks that have not been selected are still read as part of the margin read sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks does not result in a faster margin read execution time.

   Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

   NOTE

   It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

   NOTE

   Only sequential addressing is supported for margin read checks.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] bit must be set to 1.
6. Initiate margin read by setting the UT0[MRE] bit to 1.
7. Select either one's margin or zero's margin by setting the UT0[MRV] bit to the appropriate value.
8. Set the UT0[AIE] bit.

During margin read operations, with UT0[AID] low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

9. Wait until the UT0[AID] bit goes high, indicating that margin read test is complete.

   If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Step 8 in section Array integrity self check for more information.

10. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

11. Write a logic 0 to the UT0[AIE] bit.

39.6 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 40
Cross-Triggering Unit (CTU)

40.1 Chip-specific CTU information

40.1.1 CTU Configuration

The device contains one instance of Cross Triggering Unit CTU_0.

40.1.2 Features

The ADC cross-triggering unit allows automatic generation of ADC conversion requests on user selected conditions without CPU load during the PWM period and with minimized CPU load for dynamic configuration.

It implements the following features:

- Cross triggering between ADC, FlexPWM, eTimer and external Pins
- Double buffered trigger generation unit with up to 8 independent triggers generated from external triggers
- Trigger generation unit configurable in sequential mode or in triggered mode
- Trigger delay unit to compensate the delay of external low pass filter
- Double buffered global trigger unit allowing eTimer synchronization and/or ADC command generation
- Double buffered ADC command list pointers to minimize ADC-trigger unit update
- Double buffered ADC conversion command list with up to 24 ADC commands
- Each trigger has the capability to generate consecutive commands
- ADC conversion command allows to control ADC channel from each ADC, single or synchronous sampling, independent result queue selection
- DMA support with safety features

NOTE

This device does not support eTimer1 pulse and eTimer4 pulse outputs of scheduler subunit.

40.1.3 CTU inter-module connections

The following table lists the inter-module connection of the CTU for the Motor Control related peripherals for the device.

Table 40-1. Motor Control peripheral interconnects

Input Module	Input Signal	Driving Module	Driver Signal
CTU_0	PWM_REL	FlexPWM_0	Master_Reload
CTU_0	PWM_ODD_0	FlexPWM_0	Out_Trigger0_0
CTU_0	PWM_ODD_1	FlexPWM_0	Out_Trigger0_1
CTU_0	PWM_ODD_2	FlexPWM_0	Out_Trigger0_2
CTU_0	PWM_ODD_3	FlexPWM_0	Out_Trigger0_3
CTU_0	PWM_EVEN_0	FlexPWM_0	Out_Trigger1_0
CTU_0	PWM_EVEN_1	FlexPWM_0	Out_Trigger1_1
CTU_0	PWM_EVEN_2	FlexPWM_0	Out_Trigger1_2
CTU_0	PWM_EVEN_3	FlexPWM_0	Out_Trigger1_3
CTU_0	RPWM_0	FlexPWM_0	PWMX_0
CTU_0	RPWM_1	FlexPWM_0	PWMX_1
CTU_0	RPWM_2	FlexPWM_0	PWMX_2
CTU_0	RPWM_3	FlexPWM_0	PWMX_3
CTU_0	NEXT_CMD_0	ADC_0	NEX_CMD
CTU_0	FIFO_0	ADC_0	OUT
CTU_0	NEXT_CMD_1	ADC_1	NEX_CMD
CTU_0	FIFO_1	ADC_1	OUT
CTU_0	ETMR0_IN	eTimer_2	T2
CTU_0	ETMR1_IN	eTimer_1	T2
ADC_0	TRIGGER	CTU_0	TRIGGER_0
ADC_0	CMD	CTU_0	CMD_0
ADC_1	TRIGGER	CTU_0	TRIGGER_1
ADC_1	CMD	CTU_0	CMD_1

40.1.4 TGSISR input assignments for CTU_0

The following table identifies the alignment among the following:

- The input number used in the InRE and InFE fields of the CTU's register TGSISR

- The generic name used in the CTU chapter for the CTU input with that input number
- For CTU_0, the specific source on this device for the CTU input with that input number

Table 40-2. TGSISR input assignments for CTU_0

TGISR I/P	CTU generic input name	Source	Signal
0	PWM_REL	FlexPWM_0	Master_Reload
1	PWM_ODD_0	FlexPWM_0	Out_Trigger0_0
2	PWM_ODD_1	FlexPWM_0	Out_Trigger0_1
3	PWM_ODD_2	FlexPWM_0	Out_Trigger0_2
4	PWM_ODD_3	FlexPWM_0	Out_Trigger0_3
5	PWM_EVEN_0	FlexPWM_0	Out_Trigger1_0
6	PWM_EVEN_1	FlexPWM_0	Out_Trigger1_1
7	PWM_EVEN_2	FlexPWM_0	Out_Trigger1_2
8	PWM_EVEN_3	FlexPWM_0	Out_Trigger1_3
9	RPWM_0	FlexPWM_0	PWMX_0
10	RPWM_1	FlexPWM_0	PWMX_1
11	RPWM_2	FlexPWM_0	PWMX_2
12	RPWM_3	FlexPWM_0	PWMX_3
13	ETMR0_IN	eTimer_2	T2
14	ETMR1_IN	eTimer_1	T2
15	EXT_IN	External Pin	External Pin

NOTE

In above table, ETMR0_IN and ETMR1_IN generic inputs corresponds to ETMR1_IN and ETMR2_IN inputs in Figure 40-1.

40.2 Introduction

In PWM driven systems it is important to schedule the acquisition of the state variables with respect to the PWM cycle. State variables are obtained through the following peripherals: ADC, position counter (for example, quadrature decoder, resolver, and sine-cos sensor), and PWM duty cycle decoder.

The Cross-Triggering Unit (CTU) is intended to completely avoid CPU involvement in the acquisition of state variables during a control cycle defined by the PWM cycle, the half PWM cycle, or a multiple number of PWM cycles. Several CTU registers are double-buffered, thus allowing coherent update during the transition from one control cycle to the next. Thus double-buffered registers updated at control cycle N will be updated during the transition from control cycle N to control cycle N+1.

The CTU provides four FIFO memory blocks to temporarily store ADC results for CPU access.

40.3 Block diagram

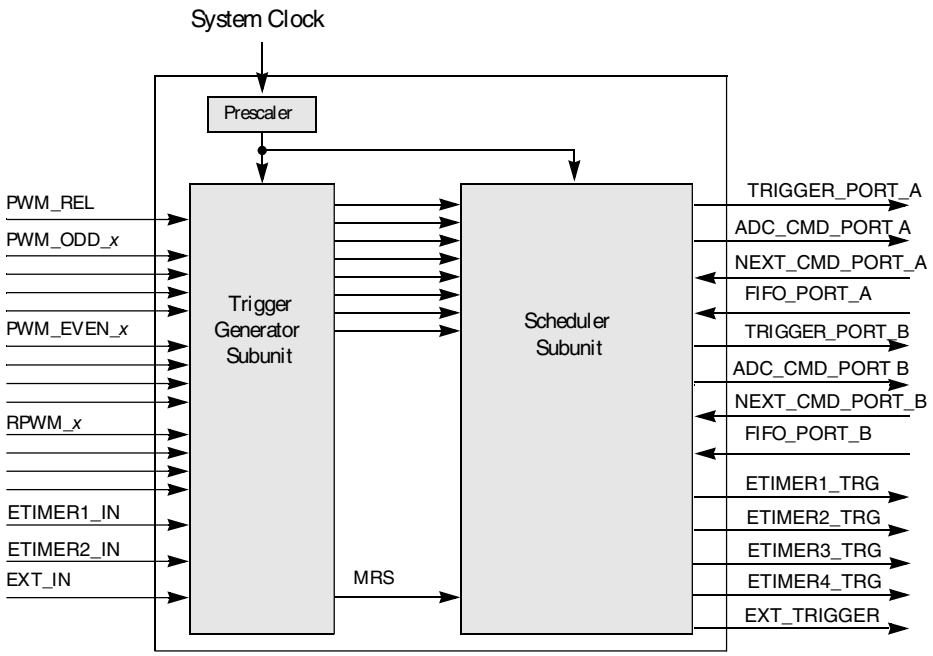


Figure 40-1. CTU block diagram

40.4 CTU overview

The CTU receives several signals from different sources such as PWM, timers, position decoder, and/or external pins. These signals are then processed to generate up to eight trigger events. An input event can be a rising edge, a falling edge, or both edges of the incoming signal. The output can be a pulse, an ADC command, or a stream of consecutive ADC commands for oversampling support. The outputs are targeted to one or more peripherals such as ADCs and timers.

The CTU implements the following features:

- Cross triggering between ADC, PWM timers, on-chip general purpose timers, and external pins
- Double-buffered trigger unit that generates up to 8 triggers
- Trigger generation unit configurable in sequential or triggered modes
- Trigger delay functionality to compensate for the delay of external low pass filters

- Double-buffered ADC command list pointers
- Double-buffered ADC conversion command list
- Each trigger has the capability to generate consecutive conversion commands
- ADC conversion commands allow two ADCs to be controlled synchronously with queue selection to store conversion results
- DMA support

In a typical application, the CTU interfaces with the following peripherals:

- PWM generators for motor control
- Timers
- External signals

The inputs are 16 digital signals. The CTU detects the rising and/or falling edges for each one of them.

Figure 40-1 shows a high level CTU block diagram. The CTU hardware architecture consists of two subunits:

- Trigger generator subunit
- Scheduler subunit

The Trigger generator subunit receives CTU input signals and selects the active edges in order to generate the Master Reload signal. The Master Reload is used to load several CTU double buffered registers. See sections TGS in triggered mode and TGS in sequential mode for more details about the Master Reload signal. The Trigger generator subunit also generates up to eight trigger event signals which are used by the Scheduler subunit.

The Scheduler subunit generates the trigger event output according to the trigger event signals from the Trigger generator subunit. The trigger event output starts an ADC conversion or generates output signals to trigger events for IPs in the SoC. See section Scheduler subunit (SU) for more detail about the Scheduler subunit.

40.4.1 Modes of operation

The CTU has two main modes of operation:

1. Triggered mode: the input event from the CTU interface is used to generate a sequence of up to 8 triggers to several outputs such as the ADCs, timers, and external triggers. Internal sequencer logic is used to schedule the triggers based upon the input event occurrence.

2. Sequential mode: each input event generates only one trigger for the selected output, such as ADCs, timers, and external triggers.

40.5 Signal description

The following table shows the chip-level signals for the CTU module.

Table 40-3. Signal properties

Name	Function
EXT_IN	Input pin for external triggers
EXT_TRG	Output pin for external triggers

40.6 Memory Map and Registers

CTU registers are implemented in two sizes:

- 32-bit registers: byte access for write operations and 32-bit access for read operation
- 16-bit registers: byte access for write operations and 32-bit access for read operations

If a 32-bit write operation is performed over a 16-bit register, the write operation is performed on the 4 aligned bytes. Read operation should be performed only at 32-bit boundary.

NOTE

Reserved bit registers, when available for write operations, should be written always with logic 0. Read access does not return meaningful data from reserved bit registers. This module does not issue transfer errors when trying to access its reserved locations.

NOTE

If the system attempts to access an invalid address within the CTU memory map, such as a non-existent register, it does not assert a slave bus error signal to the system. So the user should not rely on such an error when accessing the CTU.

The following tables describe the CTU memory mapped registers.

CTU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Trigger Generator Subunit Input Selection Register (CTU_TGSISR)	32	R/W	0000_0000h	40.6.1/1461
4	Trigger Generator Subunit Control Register (CTU_TGSCR)	16	R/W	0000h	40.6.2/1465
6	Trigger Compare Register (CTU_T0CR)	16	R/W	0000h	40.6.3/1466
8	Trigger Compare Register (CTU_T1CR)	16	R/W	0000h	40.6.3/1466
A	Trigger Compare Register (CTU_T2CR)	16	R/W	0000h	40.6.3/1466
C	Trigger Compare Register (CTU_T3CR)	16	R/W	0000h	40.6.3/1466
E	Trigger Compare Register (CTU_T4CR)	16	R/W	0000h	40.6.3/1466
10	Trigger Compare Register (CTU_T5CR)	16	R/W	0000h	40.6.3/1466
12	Trigger Compare Register (CTU_T6CR)	16	R/W	0000h	40.6.3/1466
14	Trigger Compare Register (CTU_T7CR)	16	R/W	0000h	40.6.3/1466
16	TGS Counter Compare Register (CTU_TGSCCR)	16	R/W	0000h	40.6.4/1466
18	TGS Counter Reload Register (CTU_TGSCRR)	16	R/W	0000h	40.6.5/1466
1C	Commands List Control Register 1 (CTU_CLCR1)	32	R/W	0000_0000h	40.6.6/1467
20	Commands List Control Register 2 (CTU_CLCR2)	32	R/W	0000_0000h	40.6.7/1467
24	Trigger Handler Control Register 1 (CTU_THCR1)	32	R/W	0000_0000h	40.6.8/1468
28	Trigger Handler Control Register 2 (CTU_THCR2)	32	R/W	0000_0000h	40.6.9/1472
2C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_1)	16	R/W	0000h	40.6.10/ 1475
2C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_1)	16	R/W	0000h	40.6.11/ 1476
2C	Command List Register C for self-test commands (CTU_CLR_C_1)	16	R/W	0000h	40.6.12/ 1477
2E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_2)	16	R/W	0000h	40.6.10/ 1475
2E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_2)	16	R/W	0000h	40.6.11/ 1476
2E	Command List Register C for self-test commands (CTU_CLR_C_2)	16	R/W	0000h	40.6.12/ 1477
30	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_3)	16	R/W	0000h	40.6.10/ 1475
30	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_3)	16	R/W	0000h	40.6.11/ 1476
30	Command List Register C for self-test commands (CTU_CLR_C_3)	16	R/W	0000h	40.6.12/ 1477
32	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_4)	16	R/W	0000h	40.6.10/ 1475
32	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_4)	16	R/W	0000h	40.6.11/ 1476
32	Command List Register C for self-test commands (CTU_CLR_C_4)	16	R/W	0000h	40.6.12/ 1477
34	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_5)	16	R/W	0000h	40.6.10/ 1475

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_5)	16	R/W	0000h	40.6.11/ 1476
34	Command List Register C for self-test commands (CTU_CLR_C_5)	16	R/W	0000h	40.6.12/ 1477
36	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_6)	16	R/W	0000h	40.6.10/ 1475
36	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_6)	16	R/W	0000h	40.6.11/ 1476
36	Command List Register C for self-test commands (CTU_CLR_C_6)	16	R/W	0000h	40.6.12/ 1477
38	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_7)	16	R/W	0000h	40.6.10/ 1475
38	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_7)	16	R/W	0000h	40.6.11/ 1476
38	Command List Register C for self-test commands (CTU_CLR_C_7)	16	R/W	0000h	40.6.12/ 1477
3A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_8)	16	R/W	0000h	40.6.10/ 1475
3A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_8)	16	R/W	0000h	40.6.11/ 1476
3A	Command List Register C for self-test commands (CTU_CLR_C_8)	16	R/W	0000h	40.6.12/ 1477
3C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_9)	16	R/W	0000h	40.6.10/ 1475
3C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_9)	16	R/W	0000h	40.6.11/ 1476
3C	Command List Register C for self-test commands (CTU_CLR_C_9)	16	R/W	0000h	40.6.12/ 1477
3E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_10)	16	R/W	0000h	40.6.10/ 1475
3E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_10)	16	R/W	0000h	40.6.11/ 1476
3E	Command List Register C for self-test commands (CTU_CLR_C_10)	16	R/W	0000h	40.6.12/ 1477
40	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_11)	16	R/W	0000h	40.6.10/ 1475
40	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_11)	16	R/W	0000h	40.6.11/ 1476
40	Command List Register C for self-test commands (CTU_CLR_C_11)	16	R/W	0000h	40.6.12/ 1477
42	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_12)	16	R/W	0000h	40.6.10/ 1475
42	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_12)	16	R/W	0000h	40.6.11/ 1476

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
42	Command List Register C for self-test commands (CTU_CLR_C_12)	16	R/W	0000h	40.6.12/ 1477
44	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_13)	16	R/W	0000h	40.6.10/ 1475
44	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_13)	16	R/W	0000h	40.6.11/ 1476
44	Command List Register C for self-test commands (CTU_CLR_C_13)	16	R/W	0000h	40.6.12/ 1477
46	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_14)	16	R/W	0000h	40.6.10/ 1475
46	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_14)	16	R/W	0000h	40.6.11/ 1476
46	Command List Register C for self-test commands (CTU_CLR_C_14)	16	R/W	0000h	40.6.12/ 1477
48	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_15)	16	R/W	0000h	40.6.10/ 1475
48	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_15)	16	R/W	0000h	40.6.11/ 1476
48	Command List Register C for self-test commands (CTU_CLR_C_15)	16	R/W	0000h	40.6.12/ 1477
4A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_16)	16	R/W	0000h	40.6.10/ 1475
4A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_16)	16	R/W	0000h	40.6.11/ 1476
4A	Command List Register C for self-test commands (CTU_CLR_C_16)	16	R/W	0000h	40.6.12/ 1477
4C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_17)	16	R/W	0000h	40.6.10/ 1475
4C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_17)	16	R/W	0000h	40.6.11/ 1476
4C	Command List Register C for self-test commands (CTU_CLR_C_17)	16	R/W	0000h	40.6.12/ 1477
4E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_18)	16	R/W	0000h	40.6.10/ 1475
4E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_18)	16	R/W	0000h	40.6.11/ 1476
4E	Command List Register C for self-test commands (CTU_CLR_C_18)	16	R/W	0000h	40.6.12/ 1477
50	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_19)	16	R/W	0000h	40.6.10/ 1475
50	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_19)	16	R/W	0000h	40.6.11/ 1476
50	Command List Register C for self-test commands (CTU_CLR_C_19)	16	R/W	0000h	40.6.12/ 1477

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
52	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_20)	16	R/W	0000h	40.6.10/ 1475
52	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_20)	16	R/W	0000h	40.6.11/ 1476
52	Command List Register C for self-test commands (CTU_CLR_C_20)	16	R/W	0000h	40.6.12/ 1477
54	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_21)	16	R/W	0000h	40.6.10/ 1475
54	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_21)	16	R/W	0000h	40.6.11/ 1476
54	Command List Register C for self-test commands (CTU_CLR_C_21)	16	R/W	0000h	40.6.12/ 1477
56	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_22)	16	R/W	0000h	40.6.10/ 1475
56	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_22)	16	R/W	0000h	40.6.11/ 1476
56	Command List Register C for self-test commands (CTU_CLR_C_22)	16	R/W	0000h	40.6.12/ 1477
58	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_23)	16	R/W	0000h	40.6.10/ 1475
58	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_23)	16	R/W	0000h	40.6.11/ 1476
58	Command List Register C for self-test commands (CTU_CLR_C_23)	16	R/W	0000h	40.6.12/ 1477
5A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_24)	16	R/W	0000h	40.6.10/ 1475
5A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_24)	16	R/W	0000h	40.6.11/ 1476
5A	Command List Register C for self-test commands (CTU_CLR_C_24)	16	R/W	0000h	40.6.12/ 1477
6C	FIFO DMA Control Register (CTU_FDCR)	16	R/W	0F00h	40.6.13/ 1478
70	FIFO Control Register (CTU_FCR)	32	R/W	0000_0000h	40.6.14/ 1480
74	FIFO Threshold Register (CTU_FTH)	32	R/W	0000_0000h	40.6.15/ 1482
7C	FIFO Status Register (CTU_FST)	32	R	0000_2222h	40.6.16/ 1483
80	FIFO Right Aligned Data Register (CTU_FR0)	32	R	0000_0000h	40.6.17/ 1485
84	FIFO Right Aligned Data Register (CTU_FR1)	32	R	0000_0000h	40.6.17/ 1485
88	FIFO Right Aligned Data Register (CTU_FR2)	32	R	0000_0000h	40.6.17/ 1485

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C	FIFO Right Aligned Data Register (CTU_FR3)	32	R	0000_0000h	40.6.17/ 1485
A0	FIFO Signed Left Aligned Data Register (CTU_FL0)	32	R	0000_0000h	40.6.18/ 1486
A4	FIFO Signed Left Aligned Data Register (CTU_FL1)	32	R	0000_0000h	40.6.18/ 1486
A8	FIFO Signed Left Aligned Data Register (CTU_FL2)	32	R	0000_0000h	40.6.18/ 1486
AC	FIFO Signed Left Aligned Data Register (CTU_FL3)	32	R	0000_0000h	40.6.18/ 1486
C0	Error Flag Register (CTU_EFR)	16	w1c	0000h	40.6.19/ 1487
C2	Interrupt Flag Register (CTU_IFR)	16	w1c	01FEh	40.6.20/ 1489
C4	Interrupt/DMA Register (CTU_IR)	16	R/W	0000h	40.6.21/ 1490
C6	Control ON Time Register (CTU_COTR)	16	R/W	0000h	40.6.22/ 1492
C8	Control Register (CTU_CR)	16	R/W	0000h	40.6.23/ 1492
CA	Digital Filter Register (CTU_DFR)	16	R/W	0000h	40.6.24/ 1494
CC	Expected Value A Register (CTU_EXPAR)	16	R/W	FFFFh	40.6.25/ 1495
CE	Expected Value B Register (CTU_EXPBR)	16	R/W	FFFFh	40.6.26/ 1495
D0	Counter Range Register (CTU_CNTRNGR)	16	R/W	0000h	40.6.27/ 1496
D4	List Control/Status Register (CTU_LISTCSR)	32	R/W	0000_0000h	40.6.28/ 1497

40.6.1 Trigger Generator Subunit Input Selection Register (CTU_TGSISR)

The TGSISR register is double buffered. The load from the corresponding buffer to the register is controlled by the TGSISR_RE bit in the CTU_CR register. The TGSISR_RE bit is self negated. Once set, this bit remains set until the TGSISR register is updated from the corresponding buffer register. Reads to the TGSISR register actually return the content of the buffer and not the content of the register. Thus, a read will return the most recent data written to the register address even though the buffer data is not synchronized with the register data.

NOTE

For the particular input connection assigned to each field in this register, see the CTU's chip-specific configuration details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	I15_FE	I15_RE	I14_FE	I14_RE	I13_FE	I13_RE	I12_FE	I12_RE	I11_FE	I11_RE	I10_FE	I10_RE	I9_FE	I9_RE	I8_FE	I8_RE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	I7_FE	I7_RE	I6_FE	I6_RE	I5_FE	I5_RE	I4_FE	I4_RE	I3_FE	I3_RE	I2_FE	I2_RE	I1_FE	I1_RE	I0_FE	I0_RE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSISR field descriptions

Field	Description
0 I15_FE	Input 15 Falling Edge Enable. 0 Disabled 1 Enabled
1 I15_RE	Input 15 Rising Edge Enable. 0 Disabled 1 Enabled
2 I14_FE	Input 14 Falling Edge Enable. 0 Disabled 1 Enabled
3 I14_RE	Input 14 Rising Edge Enable. 0 Disabled 1 Enabled
4 I13_FE	Input 13 Falling Edge Enable. 0 Disabled 1 Enabled
5 I13_RE	Input 13 Rising Edge Enable. 0 Disabled 1 Enabled
6 I12_FE	Input 12 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
7 I12_RE	Input 12 Rising Edge Enable. 0 Disabled 1 Enabled
8 I11_FE	Input 11 Falling Edge Enable. 0 Disabled 1 Enabled
9 I11_RE	Input 11 Rising Edge Enable. 0 Disabled 1 Enabled
10 I10_FE	Input 10 Falling Edge Enable. 0 Disabled 1 Enabled
11 I10_RE	Input 10 Rising Edge Enable. 0 Disabled 1 Enabled
12 I9_FE	Input 9 Falling Edge Enable. 0 Disabled 1 Enabled
13 I9_RE	Input 9 Rising Edge Enable. 0 Disabled 1 Enabled
14 I8_FE	Input 8 Falling Edge Enable. 0 Disabled 1 Enabled
15 I8_RE	Input 8 Rising Edge Enable. 0 Disabled 1 Enabled
16 I7_FE	Input 7 Falling Edge Enable. 0 Disabled 1 Enabled
17 I7_RE	Input 7 Rising Edge Enable. 0 Disabled 1 Enabled
18 I6_FE	Input 6 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
19 I6_RE	Input 6 Rising Edge Enable. 0 Disabled 1 Enabled
20 I5_FE	Input 5 Falling Edge Enable. 0 Disabled 1 Enabled
21 I5_RE	Input 5 Rising Edge Enable. 0 Disabled 1 Enabled
22 I4_FE	Input 4 Falling Edge Enable. 0 Disabled 1 Enabled
23 I4_RE	Input 4 Rising Edge Enable. 0 Disabled 1 Enabled
24 I3_FE	Input 3 Falling Edge Enable. 0 Disabled 1 Enabled
25 I3_RE	Input 3 Rising edge Enable 0 Disabled 1 Enabled
26 I2_FE	Input 2 Falling Edge Enable. 0 Disabled 1 Enabled
27 I2_RE	Input 2 Rising Edge Enable. 0 Disabled 1 Enabled
28 I1_FE	Input 1 Falling Edge Enable. 0 Disabled 1 Enabled
29 I1_RE	Input 1 Rising edge Enable 0 Disabled 1 Enabled
30 I0_FE	Input 0 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
31 I0_RE	Input 0 Rising Edge Enable. 0 Disabled 1 Enabled

40.6.2 Trigger Generator Subunit Control Register (CTU_TGSCR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read	0							ET_TM
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	PRES		MRS_SM					TGS_M
Reset	0	0	0	0	0	0	0	0

CTU_TGSCR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ET_TM	Enable Toggle Mode for external Trigger. 0 Pulse mode 1 Toggle mode
8–9 PRES	Prescaler selection bits for TGS and SU. This prescaler is used by the TGS counter. 00 Value is 1 01 Value is 2 10 Value is 3 11 Value is 4
10–14 MRS_SM	MRS Selection in Sequential Mode (5 bits to select one of the 32 inputs shown in the Trigger generator subunit input selection (TGSISR) register.
15 TGS_M	Trigger Generator Subunit Mode. 0 Triggered Mode 1 Sequential Mode

40.6.3 Trigger Compare Register (CTU_T*n*CR)

Address: 0h base + 6h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_T*n*CR field descriptions

Field	Description
0–15 TCRV	Trigger Compare Register Value.

40.6.4 TGS Counter Compare Register (CTU_TGSCCR)

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TGSCCV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSCCR field descriptions

Field	Description
0–15 TGSCCV	TGS Counter Compare Value. The value in this register is compared against the TGS counter. When a match occurs, the TGS counter is stopped.

40.6.5 TGS Counter Reload Register (CTU_TGSCRR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TGSCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSCRR field descriptions

Field	Description
0–15 TGSCRV	TGS Counter Reload Value. When an MRS occurs, the value in this register is used to reload the TGS counter.

40.6.6 Commands List Control Register 1 (CTU_CLCR1)

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			T3_INDEX					0			T2_INDEX					0			T1_INDEX					0			T0_INDEX				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLCR1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 T3_INDEX	Trigger 3 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 T2_INDEX	Trigger 2 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 T1_INDEX	Trigger 1 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 T0_INDEX	Trigger 0 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.

40.6.7 Commands List Control Register 2 (CTU_CLCR2)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			T7_INDEX					0			T6_INDEX					0			T5_INDEX					0			T4_INDEX				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLCR2 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 T7_INDEX	Trigger 7 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CTU_CLCR2 field descriptions (continued)

Field	Description
11–15 T6_INDEX	Trigger 6 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 T5_INDEX	Trigger 5 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 T4_INDEX	Trigger 4 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.

40.6.8 Trigger Handler Control Register 1 (CTU_THCR1)

The THCR1 register is double buffered. The update from the corresponding buffer register is controlled by the MRS signal. The THCR1 and THCR2 registers are together organized into 8 groups of 7 enable bits. Each group is associated with one trigger from the Trigger subunit. Each group of enable bits has 7 enables corresponding to a master trigger enable, External Trigger output enable, 4 eTimer outputs, and the ADC command list enable, respectively. If the master trigger enable Tn_E is cleared, the trigger is disabled and the other trigger enable bits in the group have no effect.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T3_E	T3_ ETE	T3_ T4E	T3_ T3E	T3_ T2E	T3_ T1E	T3_ADCE	0	T2_E	T2_ ETE	T2_ T4E	T2_ T3E	T2_ T2E	T2_ T1E	T2_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T1_E	T1_ ETE	T1_ T4E	T1_ T3E	T1_ T2E	T1_ T1E	T1_ADCE	0	T0_E	T0_ ETE	T0_ T4E	T0_ T3E	T0_ T2E	T0_ T1E	T0_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_THCR1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 T3_E	Trigger 3 Enable. 0 Disabled 1 Enabled
2 T3_ETE	Trigger 3 External Trigger output Enable. 0 Disabled 1 Enabled
3 T3_T4E	Trigger 3 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
4 T3_T3E	Trigger 3 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
5 T3_T2E	Trigger 3 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
6 T3_T1E	Trigger 3 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
7 T3_ADCE	Trigger 3 ADC command output Enable. 0 Disabled 1 Enabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 T2_E	Trigger 2 Enable. 0 Disabled 1 Enabled
10 T2_ETE	Trigger 2 External Trigger output Enable. 0 Disabled 1 Enabled
11 T2_T4E	Trigger 2 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
12 T2_T3E	Trigger 2 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR1 field descriptions (continued)

Field	Description
13 T2_T2E	Trigger 2 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
14 T2_T1E	Trigger 2 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
15 T2_ADCE	Trigger 2 ADC command output Enable. 0 Disabled 1 Enabled
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 T1_E	Trigger 1 Enable. 0 Disabled 1 Enabled
18 T1_ETE	Trigger 1 External Trigger output Enable. 0 Disabled 1 Enabled
19 T1_T4E	Trigger 1 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
20 T1_T3E	Trigger 1 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
21 T1_T2E	Trigger 1 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
22 T1_T1E	Trigger 1 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
23 T1_ADCE	Trigger 1 ADC command output Enable. 0 Disabled 1 Enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 T0_E	Trigger 0 Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR1 field descriptions (continued)

Field	Description
26 T0_ETE	Trigger 0 External Trigger output Enable. 0 Disabled 1 Enabled
27 T0_T4E	Trigger 0 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
28 T0_T3E	Trigger 0 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
29 T0_T2E	Trigger 0 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
30 T0_T1E	Trigger 0 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
31 T0_ADCE	Trigger 0 ADC command output Enable. 0 Disabled 1 Enabled

40.6.9 Trigger Handler Control Register 2 (CTU_THCR2)

The THCR2 register is double buffered. The update from the corresponding buffer register is controlled by the MRS signal. The THCR1 and THCR2 registers are together organized into 8 groups of 7 enable bits. Each group is associated with one trigger from the Trigger subunit. Each group of enable bits has 7 enables corresponding to a master trigger enable, External Trigger output enable, 4 eTimer outputs, and the ADC command list enable, respectively. If the master trigger enable Tn_E is cleared, the trigger is disabled and the other trigger enable bits in the group have no effect.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T7_E	T7_ ETE	T7_ T4E	T7_ T3E	T7_ T2E	T7_ T1E	T7_ADCE	0	T6_E	T6_ ETE	T6_ T4E	T6_ T3E	T6_ T2E	T6_ T1E	T6_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T5_E	T5_ ETE	T5_ T4E	T5_ T3E	T5_ T2E	T5_ T1E	T5_ADCE	0	T4_E	T4_ ETE	T4_ T4E	T4_ T3E	T4_ T2E	T4_ T1E	T4_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_THCR2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 T7_E	Trigger 7 Enable. 0 Disabled 1 Enabled
2 T7_ETE	Trigger 7 External Trigger output Enable. 0 Disabled 1 Enabled
3 T7_T4E	Trigger 7 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
4 T7_T3E	Trigger 7 ETIMER3_TRG output Enable.

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
5 T7_T2E	Trigger 7 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
6 T7_T1E	Trigger 7 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
7 T7_ADCE	Trigger 7 ADC command output Enable. 0 Disabled 1 Enabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 T6_E	Trigger 6 Enable. 0 Disabled 1 Enabled
10 T6_ETE	Trigger 6 External Trigger output Enable. 0 Disabled 1 Enabled
11 T6_T4E	Trigger 6 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
12 T6_T3E	Trigger 6 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
13 T6_T2E	Trigger 6 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
14 T6_T1E	Trigger 6 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
15 T6_ADCE	Trigger 6 ADC command output Enable. 0 Disabled 1 Enabled
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 T5_E	Trigger 5 Enable.

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
18 T5_ETE	Trigger 5 External Trigger output Enable. 0 Disabled 1 Enabled
19 T5_T4E	Trigger 5 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
20 T5_T3E	Trigger 5 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
21 T5_T2E	Trigger 5 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
22 T5_T1E	Trigger 5 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
23 T5_ADCE	Trigger 5 ADC command output Enable. 0 Disabled 1 Enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 T4_E	Trigger 4 Enable. 0 Disabled 1 Enabled
26 T4_ETE	Trigger 4 External Trigger output Enable. 0 Disabled 1 Enabled
27 T4_T4E	Trigger 4 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
28 T4_T3E	Trigger 4 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
29 T4_T2E	Trigger 4 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
30 T4_T1E	Trigger 4 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
31 T4_ADCE	Trigger 4 ADC command output Enable. 0 Disabled 1 Enabled

40.6.10 Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_*n*)

The Command List Register (CLR_x_n), where n = 1,.....,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its CMS and ST0 bits. If CMS=0 and ST0=0, the single conversion mode is selected and the CLR register assumes the CLR_A_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CIR	LC	CMS	FIFO			ST0	0			SU	0	CH			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLR_A_*n* field descriptions

Field	Description
0 CIR	Command execution Interrupt Request enable bit. 0 Disabled 1 Enabled
1 LC	Last Command bit. 0 Not last 1 Last
2 CMS	Conversion Mode Selection. Must be 0 in this register format. 0 Single conversion mode 1 Dual conversion mode
3–5 FIFO	FIFO used for ADC Port A / Port B. See CLR_A[SU] bit for Port A versus Port B selection. 000 FIFO_0 select ---- ---------- 111 FIFO_7 select

Table continues on the next page...

CTU_CLR_A_*n* field descriptions (continued)

Field	Description
6 ST0	Self Test mode control 0. Must be 0 in this register format.
7–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SU	ADC Port A / Port B selection. 0 ADC Port A selected 1 ADC Port B selected
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CH	ADC Port channel number.

40.6.11 Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_*n*)

The Command List Register (CLR_x_n), where n = 1,.....,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its CMS and ST0 bits. If CMS=1 and ST0=0, the dual conversion mode is selected and the CLR register assumes the CLR_B_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CIR	LC	CMS	FIFO			ST0		CH_B			0	CH_A			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLR_B_*n* field descriptions

Field	Description
0 CIR	Command execution Interrupt Request enable bit. 0 Disabled 1 Enabled
1 LC	Last Command bit. 0 Not last 1 Last
2 CMS	Conversion Mode Selection. Must be 1 in this register format. 0 Single conversion mode 1 Dual conversion mode

Table continues on the next page...

CTU_CLR_B_*n* field descriptions (continued)

Field	Description
3–5 FIFO	FIFO used for ADC Port A / Port B. See CLR_A[SU] bit for Port A versus Port B selection. 000 FIFO_0 select --- ---------- 111 FIFO_7 select
6 ST0	Sel-Test mode control 0. Must be 0 in this register format.
7–10 CH_B	ADC Port B channel number.
11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CH_A	ADC Port A channel number.

40.6.12 Command List Register C for self-test commands (CTU_CLR_C_*n*)

The Command List Register (CLR_x_n), where n = 1,.....,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its ST0 bit. You must first set ST0 to 1 to set the CLR register in the self-test mode register format. In this case, the CLR register assumes the CLR_C_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7
Read Write	CIR	LC	ST1	ST_CMS	ST_SU	0	ST0	ALG
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ALG	BSIZE						
Reset	0	0	0	0	0	0	0	0

CTU_CLR_C_*n* field descriptions

Field	Description
0 CIR	Command Execution Interrupt Request enable bit (this bit has no influence in this mode). **NOTE:** CIR bit value has no influence in self-test mode. The interrupt request is disabled and cannot be enabled by using this bit in this mode.

Table continues on the next page...

CTU_CLR_C_*n* field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
1 LC	Last command bit. 0 Not last 1 Last
2 ST1	Self Test command control 1. ST1 must be 0 for a self test command.
3 ST_CMS	Conversion Mode Selection for Self Test command. 0 Single Conversion Mode 1 Dual Conversion Mode
4 ST_SU	Self Test Selection Unit bit. Usable only if ST_CMS is 0. 0 ADC Port A 1 ADC Port B
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ST0	Self Test command control 0. ST0 must 1 in this register format.
7–8 ALG	Algorithm scheduled: 00 Algorithm S 01 Reserved 10 Algorithm C 11 Full algorithm (S + C)
9–15 BSIZE	Burst size of the algorithm iteration. 0000000 Single step execution ..... Two step execution through 511 step execution 1111111 512 step execution with one trigger

40.6.13 FIFO DMA Control Register (CTU_FDCR)

The FDCR register enables the DMA requests issued by the CTU FIFOs. The DMA request occurs when it is enabled and the corresponding FIFO had reached a threshold defined in register FTH.

NOTE

Interrupt and DMA are controlled independently, thus FIFO DMA operation and interrupts can be enabled at the same time.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7
Read	Reserved				Reserved	Reserved	Reserved	Reserved
Write	w1c				w1c	w1c	w1c	w1c
Reset	0	0	0	0	1	1	1	1

Bit	8	9	10	11	12	13	14	15
Read	Reserved				DE3	DE2	DE1	DE0
Write	w1c							
Reset	0	0	0	0	0	0	0	0

CTU_FDCR field descriptions

Field	Description
0–3 Reserved	This field is reserved.
4 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
5 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
6 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
7 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
8–11 Reserved	This field is reserved. **NOTE:** The reset value of each individual bit is 0b1. If this reset value is to be preserved during a write to these bits, and 0b0 must be written to each of these bits.
12 DE3	FIFO 3 DMA enable. 0 Disabled 1 Enabled
13 DE2	FIFO 2 DMA enable. 0 Disabled 1 Enabled
14 DE1	FIFO 1 DMA enable.

Table continues on the next page...

CTU_FDCR field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
15 DE0	FIFO 0 DMA enable. 0 Disabled 1 Enabled

40.6.14 FIFO Control Register (CTU_FCR)

This register implements interrupt enable bits to signal FIFO behavior. FIFO overrun, overflow, empty, and full indication from one of the FIFOs are OR-ed together to indicate through one interrupt line that one or more flags are active. After receiving an interrupt the FST register should be checked to evaluate what caused that interrupt. There is one interrupt line per FIFO module.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OR_EN3	OF_EN3	EMPTY_EN3	FULL_EN3	OR_EN2	OF_EN2	EMPTY_EN2	FULL_EN2	OR_EN1	OF_EN1	EMPTY_EN1	FULL_EN1	OR_EN0	OF_EN0	EMPTY_EN0	FULL_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OR_EN3	FIFO 3 Overrun interrupt enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_FCR field descriptions (continued)

Field	Description
17 OF_EN3	FIFO 3 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
18 EMPTY_EN3	FIFO 3 Empty interrupt enable. 0 Disabled 1 Enabled
19 FULL_EN3	FIFO 3 Full interrupt enable. 0 Disabled 1 Enabled
20 OR_EN2	FIFO 2 Overrun interrupt enable. 0 Disabled 1 Enabled
21 OF_EN2	FIFO 2 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
22 EMPTY_EN2	FIFO 2 Empty interrupt enable. 0 Disabled 1 Enabled
23 FULL_EN2	FIFO 2 Full interrupt enable. 0 Disabled 1 Enabled
24 OR_EN1	FIFO 1 Overrun interrupt enable. 0 Disabled 1 Enabled
25 OF_EN1	FIFO 1 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
26 EMPTY_EN1	FIFO 1 Empty interrupt enable. 0 Disabled 1 Enabled
27 FULL_EN1	FIFO 1 Full interrupt enable. 0 Disabled 1 Enabled
28 OR_EN0	FIFO 0 Overrun interrupt enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_FCR field descriptions (continued)

Field	Description
29 OF_EN0	FIFO 0 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
30 EMPTY_EN0	FIFO 0 Empty interrupt enable. 0 Disabled 1 Enabled
31 FULL_EN0	FIFO 0 Full interrupt enable. 0 Disabled 1 Enabled

40.6.15 FIFO Threshold Register (CTU_FTH)

This register defines the threshold value for four FIFO modules. The threshold is used for comparison against the FIFO pointer and to evaluate if an overflow condition occurred for that FIFO. The overflow flag indicates that a certain amount of data, defined by the threshold value, is available in the FIFO for read. The overflow condition is set if the number of elements in the FIFO is greater than the threshold value. Thus, out of reset this register is set to detect one element in the respective FIFO since the initial value is zero.

NOTE

The threshold value should not exceed the FIFO size

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TH3								TH2								TH1								TH0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FTH field descriptions

Field	Description
0–7 TH3	FIFO 3 Threshold.
8–15 TH2	FIFO 2 Threshold.
16–23 TH1	FIFO 1 Threshold.
24–31 TH0	FIFO 0 Threshold.

40.6.16 FIFO Status Register (CTU_FST)

This register has the flags that indicate the status of the FIFO modules. Up to four FIFOs are supported. FIFO overrun, overflow, empty, and full flags are available. If enabled by the FCR register bits, these flags can generate an interrupt to the CPU. There is one interrupt signal per FIFO which combines overrun, overflow, empty, and full flags.

NOTE

The only actual error indication corresponds to the overrun flags which stay asserted once the overrun condition occurs. They are cleared by writing 1 to the respective flag. The overflow, empty, and full flags correspond to the current state of the FIFOs.

NOTE

Bits OR3, OR2, OR1, and OR0 can be read and cleared by software

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OR3	OF3	EMP3	FULL3	OR2	OF2	EMP2	FULL2	OR1	OF1	EMP1	FULL1	OR0	OF0	EMP0	FULL0
W	w1c				w1c				w1c				w1c			
Reset	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

CTU_FST field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OR3	FIFO 3 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occurred
17 OF3	FIFO 3 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
18 EMP3	FIFO 3 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
19 FULL3	FIFO 3 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
20 OR2	FIFO 2 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occurred
21 OF2	FIFO 2 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
22 EMP2	FIFO 2 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
23 FULL2	FIFO 2 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
24 OR1	FIFO 1 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occurred
25 OF1	FIFO 1 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
26 EMP1	FIFO 1 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
27 FULL1	FIFO 1 Full interrupt flag.

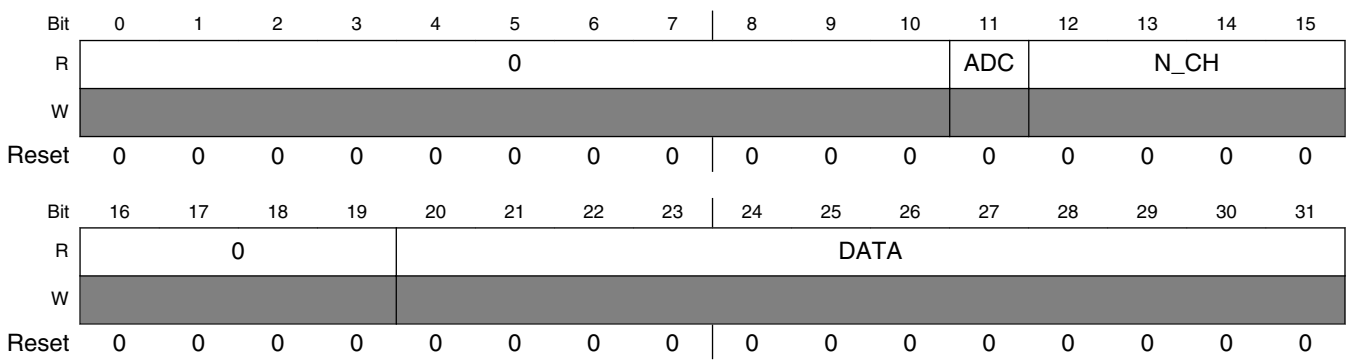
Table continues on the next page...

CTU_FST field descriptions (continued)

Field	Description
	0 Interrupt has not occurred 1 Interrupt has occurred
28 OR0	FIFO 0 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occurred
29 OF0	FIFO 0 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
30 EMP0	FIFO 0 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred
31 FULL0	FIFO 0 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occurred

40.6.17 FIFO Right Aligned Data Register (CTU_FR*n*)

Address: 0h base + 80h offset + (4d × i), where i=0d to 3d



CTU_FR*n* field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ADC	This bit indicates from which ADC Port the value in the DATA field corresponds to. 0 Data coming from ADC Port B 1 Data coming from ADC Port A
12–15 N_CH	Number of the channel that DATA field corresponds to.

Table continues on the next page...

CTU_FR*n* field descriptions (continued)

Field	Description
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA	Data of stored channel.

40.6.18 FIFO Signed Left Aligned Data Register (CTU_FL*n*)

NOTE

A read of this register consumes the data from the respective FIFO.

Address: 0h base + A0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											ADC	N_CH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LA_DATA													0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FL*n* field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ADC	This bit indicates from which ADC Port the value in the DATA field corresponds to. 0 Data coming from ADC Port B 1 Data coming from ADC Port A
12–15 N_CH	Number of the channel that DATA field corresponds to.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–28 LA_DATA	Data of stored channel.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.6.19 Error Flag Register (CTU_EFR)

The EFR register bits are cleared if a 1 is written to the bit position. Otherwise, the bit retains the previous value.

NOTE

The ERRCMP field reads 0 at reset. When reset is released, this field remains cleared until the CTU timer clock is enabled. After the CTU timer clock is enabled, ERRCMP reads 1. This field must be cleared before enabling the interrupt requests.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7
Read	0		LIST_BE	CS	ET_OE	ERRCMP	T4_OE	T3_OE
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	T2_OE	T1_OE	ADC_OE	TGS_OSM	MRS_O	ICE	SM_TO	MRS_RE
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

CTU_EFR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 LIST_BE	List Busy Error. Indicates that a list of commands is trying to send commands to a busy ADC or to the same ADC at the same time. It also indicates a conflict in the self-test algorithm execution by the ADCs, see Self-Test Execution Errors section. Cleared if 1 is written to this bit location. 0 No error 1 Error
3 CS	Self test Counter Status. This bit indicates if a self-test algorithm is in progress. If 1 is written to this bit, a reset to the self test counter will occur, thus terminating the self-test algorithm execution. As a status bit it indicates: 0 All the counters are at reset value, thus none of the self-test algorithm are executing 1 One of the counters used in self test mode is not in the reset state, thus self-test is in progress.
4 ET_OE	External Trigger generation Overrun Error. This bit is set when an external trigger occurs while a trigger input is being processed by the TGS. It is cleared if 1 is written to this bit location. 0 No error 1 Error
5 ERRCMP	Error Compare. When a match occurs, the TGS Counter has reached the TGSCCR value. Cleared if 1 is written to this bit location.

Table continues on the next page...

CTU_EFR field descriptions (continued)

Field	Description
	0 No match 1 Match
6 T4_OE	Timer 4 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
7 T3_OE	Timer 3 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
8 T2_OE	Timer 2 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
9 T1_OE	Timer 1 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
10 ADC_OE	ADC command generation Overrun Error. This bit is set if an ADC command is issued when there is a command already being processed. Cleared if 1 is written to this bit location. 0 No Error 1 Error
11 TGS_OSM	The TGS Overrun in Sequential Mode. This bit is set when a new event occurs before the trigger event selected by the previous event had occurred. The error is generated only if the trigger from the TGS is linked to a SU output. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun
12 MRS_O	Master Reload Signal Overrun. If the time window defined by two consecutive MRS is not enough for the number of programmed events in the THCR registers, an error occurs and MRS_O flag is set. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun
13 ICE	Invalid Command Error. An error that occurs when the shared channel is accessed by two ADCs at the same time, or when an ADC is already using the channel. This situation can occur with a dual conversion command or with two single conversions in parallel mode. 0 No error 1 Error
14 SM_TO	Trigger Overrun (more than 8 ES) in TGS Sequential Mode. If the number of positive edges of ES is higher than 8, an error occurs and SM_TO flag is set. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun

Table continues on the next page...

CTU_EFR field descriptions (continued)

Field	Description
15 MRS_RE	Master Reload Signal Reload Error. If an MRS occurs while the user is updating a double-buffered register, the MRS_RE is set indicating an error condition. Cleared if 1 is written to this bit location. 0 No error 1 Error

40.6.20 Interrupt Flag Register (CTU_IFR)

The IFR register bits are cleared if a 1 is written to the bit position. Otherwise, the bit retains the previous value.

NOTE

The Tn_I fields read 0 at reset. When reset is released, these fields remain cleared until the CTU timer clock is enabled. After the CTU timer clock is enabled, the Tn_I fields read 1. These flags must be cleared before enabling the interrupt requests.

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7
Read	0				SERR_B	SERR_A	ADC_I	T7_I
Write					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	1

Bit	8	9	10	11	12	13	14	15
Read	T6_I	T5_I	T4_I	T3_I	T2_I	T1_I	T0_I	MRS_I
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	1	1	1	1	1	1	1	0

CTU_IFR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SERR_B	If this bit is set, it means that the time between the start of a conversion and the end of that conversion is out of the expected range which is defined by the EXPBR and CNTRNGR registers. See Expected Value B (EXPBR) and Counter Range (CNTRNGR) registers. Cleared if 1 is written to this bit location.
5 SERR_A	If this bit is set, it means that the time between the start of a conversion and the end of that conversion is out of the expected range defined by the EXPAR and CNTRNGR registers. See the Expected value A (EXPAR) and Counter Range (CNTRNGR) registers. Cleared if 1 is written to this bit location.
6 ADC_I	ADC command interrupt flag is set when a new command is issued. The interrupt related to this flag is enabled by the CIR bit in the CLR_x_n registers. Cleared if 1 is written to this bit location.

Table continues on the next page...

CTU_IFR field descriptions (continued)

Field	Description
7 T7_I	Trigger 7 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
8 T6_I	Trigger 6 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
9 T5_I	Trigger 5 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
10 T4_I	Trigger 4 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
11 T3_I	Trigger 3 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
12 T2_I	Trigger 2 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
13 T1_I	Trigger 1 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
14 T0_I	Trigger 0 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
15 MRS_I	MRS Interrupt flag is set when the Master Reload Signal occurs. The interrupt associated with this flag is enabled by the MRS_IE bit in the IR register. Cleared if 1 is written to this bit location.

40.6.21 Interrupt/DMA Register (CTU_IR)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	T7_IE	T6_IE	T5_IE	T4_IE	T3_IE	T2_IE	T1_IE	T0_IE	0		SAF_CNT_B_EN	SAF_CNT_A_EN	DMA_DE	MRS_DMAE	MRS_IE	IEE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_IR field descriptions

Field	Description
0 T7_IE	Trigger 7 Interrupt Enable. 0 Disabled 1 Enabled
1 T6_IE	Trigger 6 Interrupt Enable. 0 Disabled 1 Enabled
2 T5_IE	Trigger 5 Interrupt Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_IR field descriptions (continued)

Field	Description
3 T4_IE	Trigger 4 Interrupt Enable. 0 Disabled 1 Enabled
4 T3_IE	Trigger 3 Interrupt Enable. 0 Disabled 1 Enabled
5 T2_IE	Trigger 2 Interrupt Enable. 0 Disabled 1 Enabled
6 T1_IE	Trigger 1 Interrupt Enable. 0 Disabled 1 Enabled
7 T0_IE	Trigger 0 Interrupt Enable. 0 Disabled 1 Enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SAF_CNT_B_EN	Enable the ADC Port B counter to check the conversion time. 0 Disabled 1 Enabled
11 SAF_CNT_A_EN	Enable the ADC Port A counter to check the conversion time. 0 Disabled 1 Enabled
12 DMA_DE	If this bit is set, a dma done acts as a write '1' in the GRE bit.
13 MRS_DMAE	If GRE bit is set, DMA request is issued on MRS occurrence.
14 MRS_IE	MRS Interrupt Enable. 0 Disabled 1 Enabled
15 IEE	Interrupt Error Enable. 0 Disabled 1 Enabled

40.6.22 Control ON Time Register (CTU_COTR)

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								COTGT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_COTR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 COTGT	Control ON-Time and Guard Time for external trigger. See Scheduler subunit (SU) for a more detailed description.

40.6.23 Control Register (CTU_CR)

NOTE

The CTU_ADC_R bit resets all lists in execution. If CTU list execution is in parallel mode, this bit resets both lists even if they issued commands to different ADCs. Also, the self-test algorithm counters go to reset state.

NOTE

Bits Tn_SG, CTU_ADC_R, CGRE, and MRS_SG can be set by software. Bits CTU_ODIS, DFE, GRE, and TGSISR_RE can be read and set by software.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7
Read								
Write	T7_SG	T6_SG	T5_SG	T4_SG	T3_SG	T2_SG	T1_SG	T0_SG
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read		CTU_ODIS	DFE		FGRE		GRE	TGSISR_RE
Write	CTU_ADC_R			CGRE		MRS_SG		
Reset	0	0	0	0	0	0	0	0

CTU_CR field descriptions

Field	Description
0 T7_SG	Trigger 7 Software Generated. 0 S/W not generated 1 S/W generated
1 T6_SG	Trigger 6 Software Generated. 0 S/W not generated 1 S/W generated
2 T5_SG	Trigger 5 Software Generated. 0 S/W not generated 1 S/W generated
3 T4_SG	Trigger 4 Software Generated. 0 S/W not generated 1 S/W generated
4 T3_SG	Trigger 3 Software Generated. 0 S/W not generated 1 S/W generated
5 T2_SG	Trigger 2 Software Generated. 0 S/W not generated 1 S/W generated
6 T1_SG	Trigger 1 Software Generated. 0 S/W not generated 1 S/W generated
7 T0_SG	Trigger 0 Software Generated. 0 S/W not generated 1 S/W generated
8 CTU_ADC_R	CTU command list control state machine Reset. Self negated bit. This bit resets the command list in the CTU in case the ADC does not respond to a conversion command within the expected time window. After a reset, the command list stops execution and waits for a new trigger event in order to initiate a command list execution.
9 CTU_ODIS	CTU Output Disable. 0 Enabled 1 Disabled
10 DFE	Digital Filter Enable. 0 Disabled 1 Enabled
11 CGRE	Clear GRE to 0.
12 FGRE	Flag GRE. See Reload mechanism for more details.

Table continues on the next page...

CTU_CR field descriptions (continued)

Field	Description
13 MRS_SG	MRS Software Generated. 0 S/W not generated 1 S/W generated
14 GRE	General Reload Enable. See Reload mechanism for more details. 0 Disabled 1 Enabled
15 TGSISR_RE	TGS Input Selection Register Reload Enable. Controls the re-load of the TGSISR register. This bit is self negated, thus it always returns zero. See Trigger Generator Subunit Input Selection Register (CTU_TGSISR) for more details.

40.6.24 Digital Filter Register (CTU_DFR)

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								FILTER_N							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_DFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 FILTER_N	Digital Filter value. The external signal (EXT_IN) is considered at 1 if it is latched FILTER_N times at 1 and is considered at 0 if it is latched FILTER_N times at 0. If FILTER_N = 0, the filter is in bypass. 00000000 Filter is in bypass 00000001 In all combinations of FILTERN_N greater than 0, the filter is not in bypass -------- ---------- 11111111 In all combinations of FILTERN_N greater than 0, the filter is not in bypass

40.6.25 Expected Value A Register (CTU_EXPAR)

The EXPAR register has the expected number of system clock cycles (the slave bus clock) for an AD conversion to be completed by the ADC connected to Port A. If SAF_CNT_A_EN is set in the IR register and a conversion takes a number of cycles out of the interval defined by EXPA and CNTRNG to complete, then the SERR_A error bit is set in the IFR register. Please see the CNTRNGR for more details.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	EXPA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CTU_EXPAR field descriptions

Field	Description
0–15 EXPA	This value is the expected number of system clock cycles needed by ADC Port A to complete a conversion.

40.6.26 Expected Value B Register (CTU_EXPBR)

The EXPBR register has the expected number of system clock cycles (the slave bus clock) for an AD conversion to be completed by the ADC connected to Port B. If SAF_CNT_B_EN is set in the IR register and a conversion takes a number of cycles out of the interval defined by EXPB and CNTRNG to complete, then the SERR_B error bit is set in the IFR register. Please see the CNTRNGR for more details.

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	EXPB															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CTU_EXPBR field descriptions

Field	Description
0–15 EXPB	This value is the expected number of system clock cycles needed by ADC Port B to complete a conversion.

40.6.27 Counter Range Register (CTU_CNTRNGR)

The CNTRNGR register provides a way to mask out the less significant bits in the EXP_A and EXP_B fields of EXPAR and EXPBR registers, respectively. If set, bits in CNTRNG mask out related bits in EXP_A and EXP_B fields, thus providing a range in which a conversion should occur. The expression used for the conversion range check is: set SERR_A bit if (conversion counter & ~CNTRNG > EXP_A & ~CNTRNG) or (conversion counter & ~CNTRNG < EXPP_A & ~CNTRNG), where the conversion counter increments at system clock rate (slave bus clock rate). This equation also applies to SERR_B and EXP_B.

NOTE

The 1s in the CNTRNG must be set from the right to the left (less significant bits) without gap with 0.

NOTE

The safety counter allowed range is obtained by taking the EXPA/B value and exchanging to "X" the corresponding positions where the CNTRNG bit is 1.

Address: 0h base + D0h offset = D0h

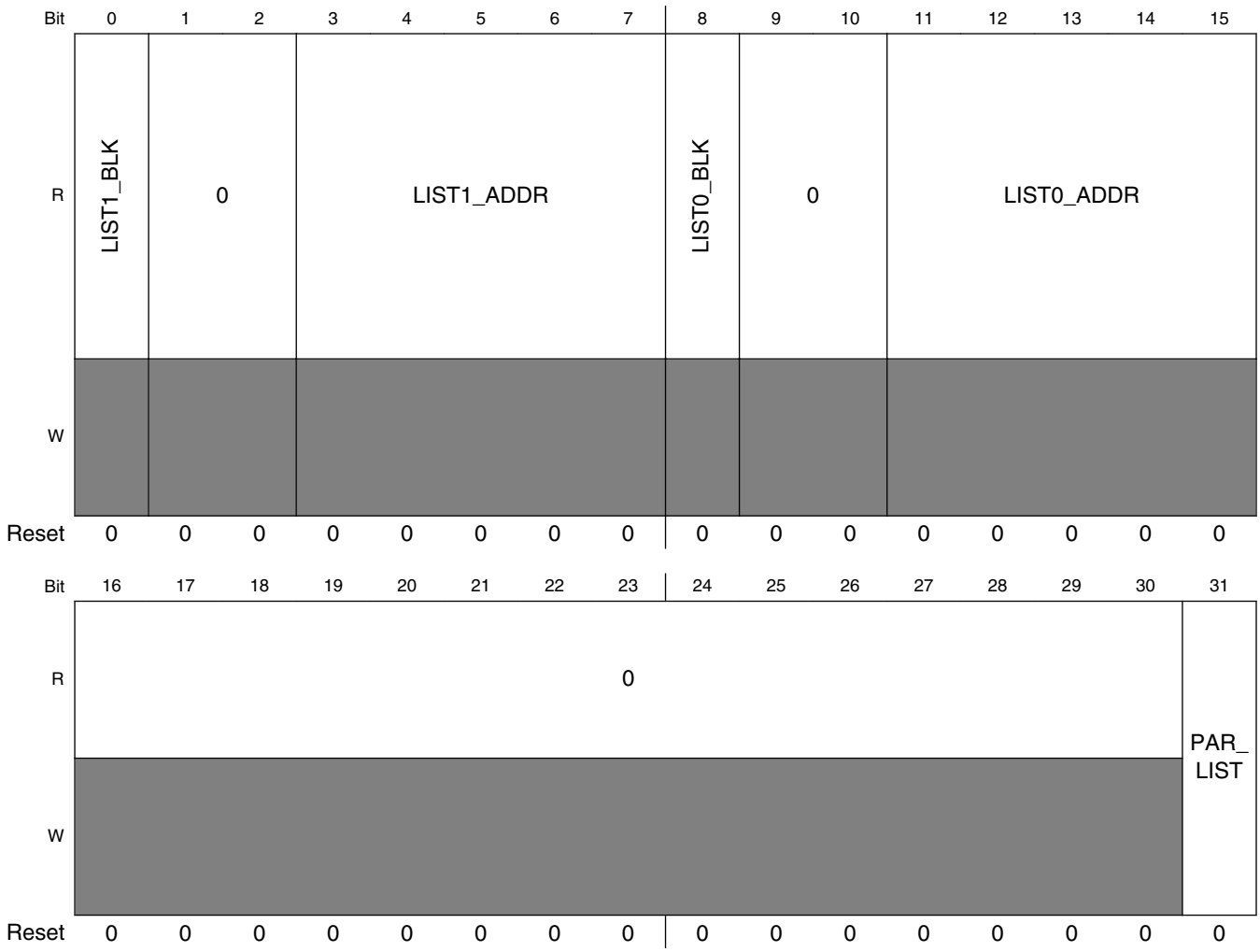
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								CNTRNG							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CNTRNGR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 CNTRNG	If set, the bits in this field mask out the equivalent bits of the expected values EXP_A and EXP_B. In this way, a range of system clock (slave bus clock) cycles is defined for a conversion to occur. This means the bit positions with 1s in the CNTRNG change the corresponding bits on EXPA/EXPB and safety counters A/B to "X". For example, with EXPA = 1b0010_1101 and CNTRNG = 0b0111, the safety counter can end with values of the format 0b0010_1XXX to avoid SERR_A error flag.

40.6.28 List Control/Status Register (CTU_LISTCSR)

Address: 0h base + D4h offset = D4h



CTU_LISTCSR field descriptions

Field	Description
0 LIST1_BLK	List 1 Blocked flag. When set, this bit indicates that the address indicated by LIST1_ADDR had failed when trying to issue a command to the corresponding ADC. This bit is only meaningful if LIST_BE flag in EFR register is set. This bit is cleared when LIST_BE flag is cleared.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 LIST1_ADDR	List Address 1. This field shows the command address that was being executed when the LIST_BE flag in the EFR register was set.
8 LIST0_BLK	List 0 Blocked flag. This bit indicates that the address indicated by LIST0_ADDR had failed when trying to issue a command to the corresponding ADC. This bit is only meaningful if LIST_BE flag in EFR register is set. This bit is cleared when LIST_BE flag is cleared.

Table continues on the next page...

CTU_LISTCSR field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 LIST0_ADDR	List Address 0. Indicates the command address being executed when LIST_BE flag in EFR register was set.
16–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PAR_LIST	Parallel mode List. This bit selects the parallel mode for the list execution. 0 List is executed in streaming mode 1 List is executed in parallel mode

40.7 Functional description

The CTU is comprised by two subunits: the Trigger Generator subunit, or TGS, and the Scheduler subunit, or SU. The TGS is responsible for generating triggers based on the module input signals connected to timer modules. The trigger signal from TGS is then connected to the scheduler subunit, which is responsible for the generation of ADC commands or triggers to on-chip logic such as timers. The following sections describe the TGS and SU in more detail.

40.7.1 Trigger Generator subunit (TGS)

The Trigger Generator subunit, TGS, generates up to eight trigger events. The main sub-blocks in the TGS are:

- Counter: generates sequential trigger events
- Trigger List: with 8 x 16-bit double-buffered registers for timing comparison

The TGS has two modes of operation:

- **Triggered mode**: each event source generates up to eight trigger event outputs.
- **Sequential mode**: each event source generates only one trigger event output.

Note

An *event* means a pulse generated from the selected CTU trigger inputs. It is represented in Figure 40-4 by the ES signal (event signal).

External signals, represented by EXT_IN in Figure 40-4, may be asynchronous related to the CTU clock. For this reason the external trigger inputs are filtered by a CTU programmable digital filter. The external signal is considered at 1 if it is stable at 1 during FILTER_N CTU clock cycles and it is considered to be at 0 if it is stable at 0 during FILTER_N CTU clock cycles, where FILTER_N is defined in the digital filter register (DFR).

Trigger events can be initiated by hardware or by software. A software trigger is generated by writing to the CTU Control Register, CR. The CR register Tx_SG bits are automatically cleared by the CTU hardware when the respective trigger event is generated.

40.7.2 TGS in triggered mode

Figure 40-2 shows the TGS configuration in Triggered mode.

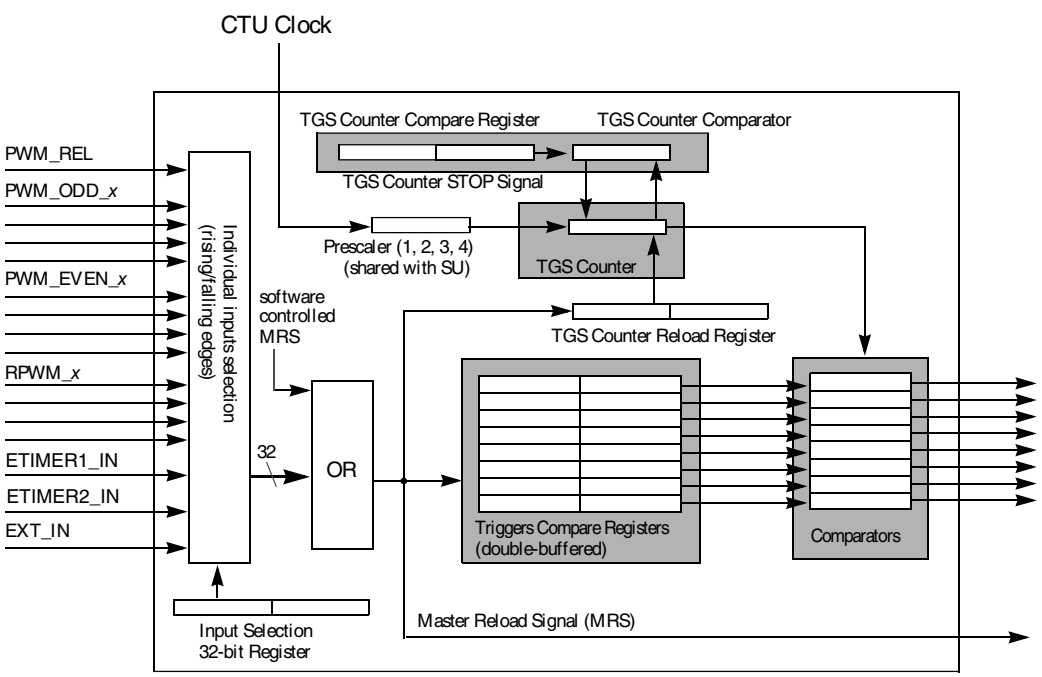


Figure 40-2. Trigger Generator subunit in triggered mode

Note

The CTU clock should be connected to the same clock source used by the CTU inputs clock. In Figure 40-2 these inputs are from the PWM generator, eTimers, and EXT_IN.

The TGS has 16 input signals selected from the input selection register (TGSISR). The available selections are:

- Inactive, no selection
- Rising edge
- Falling edge
- Both edges

Up to 32 input events corresponding to 16 input signals are selected through the TGSISR register and OR-ed in order to generate the Master Reload Signal (MRS), which defines a *control cycle*.

The MRS can also be generated by software. The software operation is done by writing to the MRS_SG bit in the CTU Control Register. The MRS_SG bit is self cleared; thus, it reads always as 0. The MRS is used to re-load the TGS Counter Register with the values from the double-buffered TGSCRR register. The MRS is also used to re-load all double-buffered registers, such as the trigger compare registers, TGSCR and TGSCRR, with the corresponding buffered values.

The triggers list registers consist of 8 compare registers (T0CR through T7CR). Each register is associated with a comparator, where a match generates an output trigger. On the occurrence of a re-load triggered by MRS, the comparison for trigger events generation starts from the first cycle of counting. However, the comparison with the TGS Counter Compare Register (TGSCCR) is always disabled during one TGS clock cycle and then re-enabled. A match between TGSCCR and the TGS counter stops the TGS counter. Therefore, this one cycle disable allows the user to configure the TGSCCR and TGSCRR with the same value.

The prescaler for TGS and SU is defined by the PRES bits in the TGS Control Register and divides the CTU clock. The available selections for the prescaler values are 1, 2, 3, and 4.

Figure 40-3 shows a timing diagram for the TGS operation in triggered mode. The MRS and the trigger event occurrences are indicated along with the delay relative to the MRS occurrence.

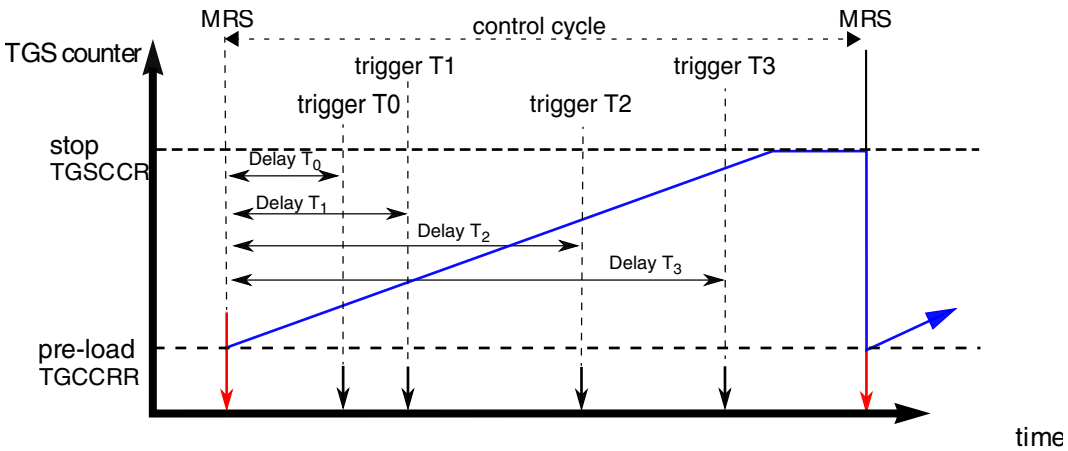


Figure 40-3. Example timing diagram for TGS in triggered mode

40.7.3 TGS in sequential mode

Figure 40-4 shows the blocks and signals related to the TGS in sequential mode.

In sequential mode only one of the 32 input signals is selected by the 5-bit MRS_SM (master reload selection). The selected signal re-loads the trigger list and resets the 3-bit ES counter which selects the trigger event. Sequences of up to eight trigger events are generated in one control cycle.

The 32 input events corresponding to 16 input signals with 2 edges detection per signal are individually enabled through the register TGSISR and OR-ed in order to generate the event signal ES. The ES signal is used to re-load the TGS counter register and to increment the 3-bit ES counter.

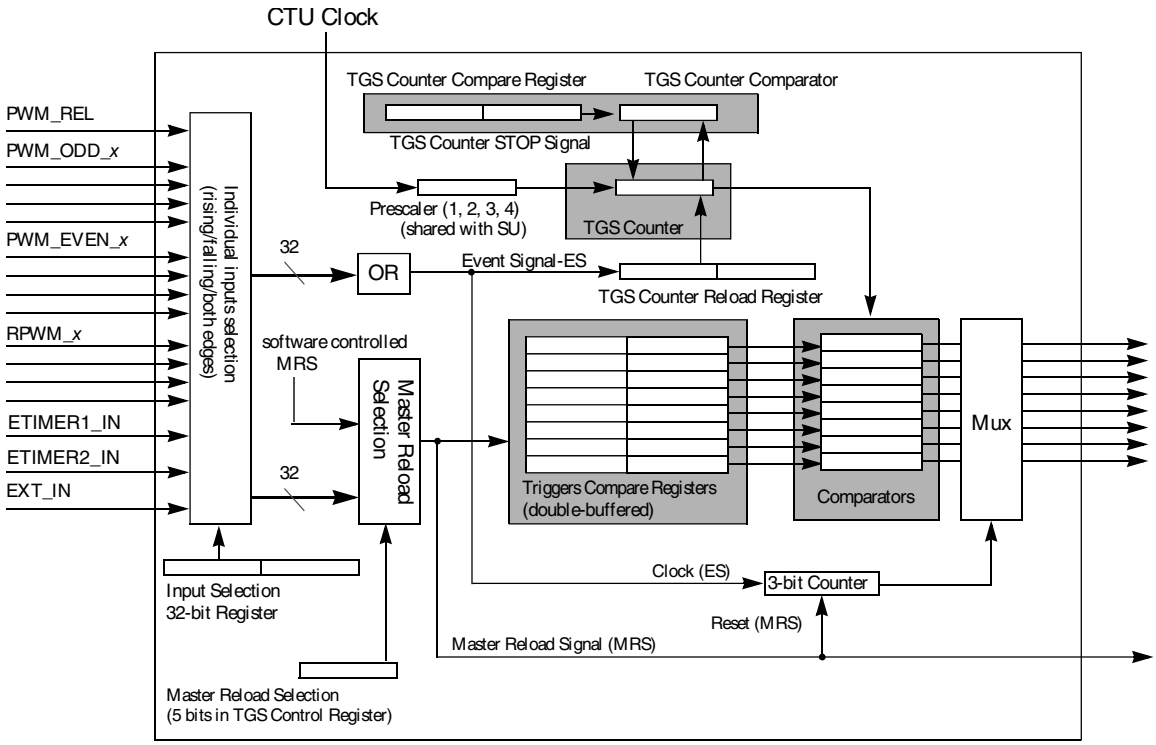


Figure 40-4. TGS in sequential mode

Figure 40-5 is a timing diagram for TGS in sequential mode showing the MRS and ES. The trigger events are indicated with the delay with respect to the ES. Note that initially ES and MRS are aligned. The TGS counter is re-loaded on each ES and starts counting up until the next ES or until it matches the value in TGSCCR register and stops.

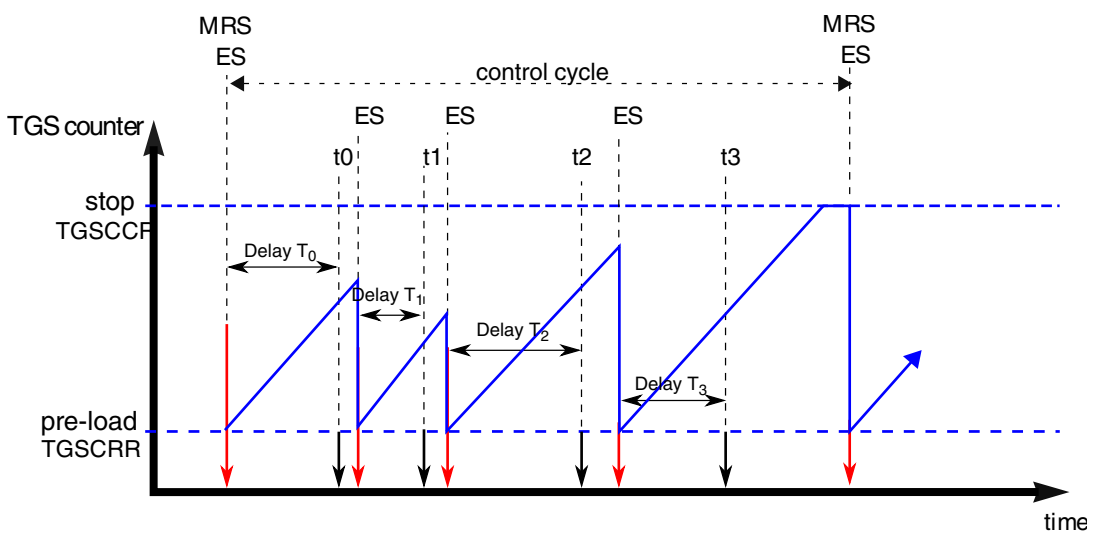


Figure 40-5. Example timing diagram for TGS in sequential mode

40.7.4 TGS counter

The TGS counter counts from negative to positive values as defined by the re-load and stop values defined in register TGSCRR and TGSCCR, respectively. The maximum counter value is 0x7FFF; after that, a counter wrap occurs and the counter value transitions from 0x7FFF to 0x8000. The following figure shows TGS counting examples.

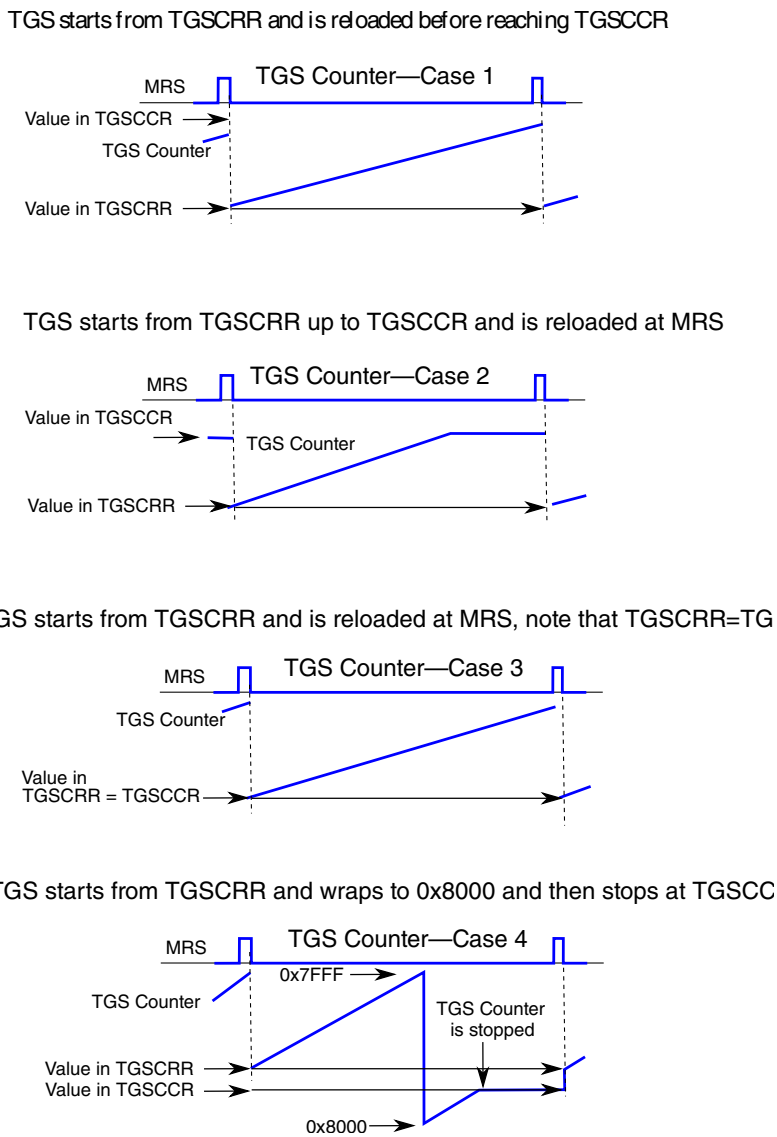


Figure 40-6. TGS counter behavior

Note

The TGS counter starts counting after an MRS and only stops if a counter wrap occurs and TGSCCR value is reached before the next MRS occurs.

40.7.5 Scheduler subunit (SU)

The following figure shows the block diagram of the Scheduler subunit. The SU trigger event outputs are:

1. ADC command or ADC stream of commands
2. eTimer1 pulse
3. eTimer2 pulse
4. eTimer3 pulse
5. eTimer4 pulse
6. External trigger pulse

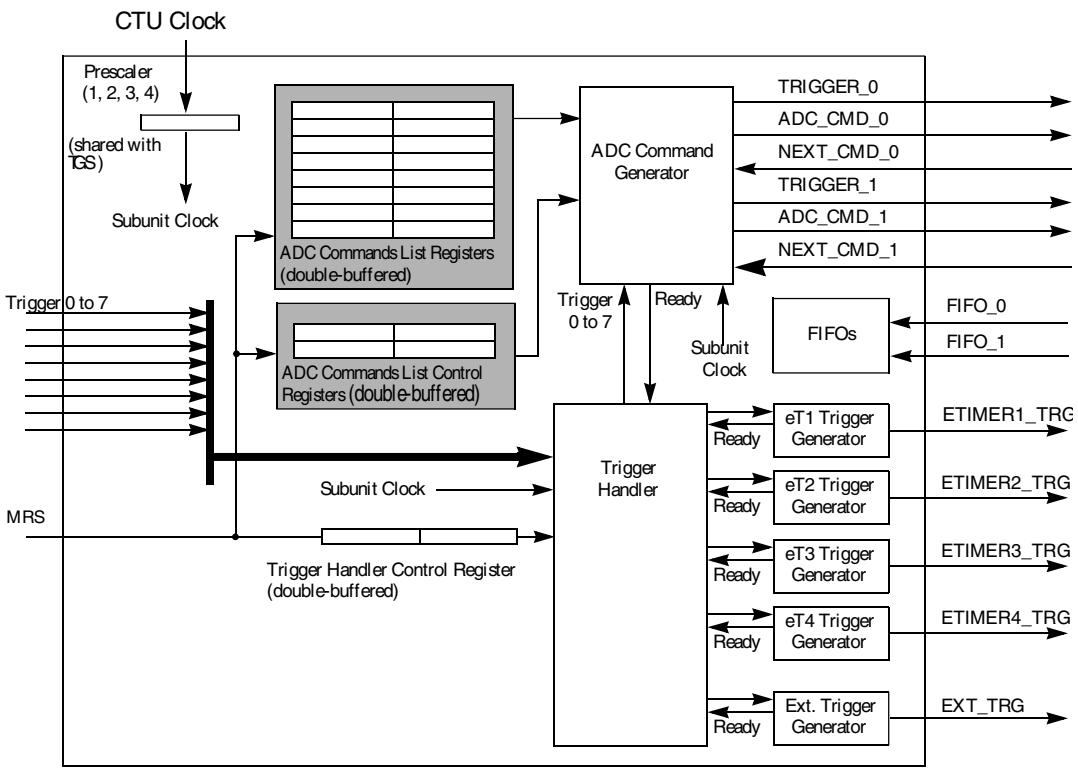


Figure 40-7. Scheduler subunit

The SU receives 8 trigger signals from the Trigger Generator subunit, TGS, and starts a command list to the selected ADC or generates the trigger event outputs. This module has the same functionality in both TGS modes: triggered and sequential. Each of the 4 SU outputs can be linked to any of the 8 trigger events from TGS. This is implemented by the Trigger Handler block. Each trigger event can be linked to one or more SU outputs.

If two events are linked at the same time to the same output (either ADC commands or output triggers) an error is generated and flagged in the EFR register's TnOE bits. In this case, only the trigger with the lowest index is used. For example, if trigger 0 and trigger 1 are linked to the same ADC output and they occur at the same time, an error is generated and the ADC command corresponding to trigger 0 is generated.

When a trigger is linked to an ADC, an associated ADC stream of commands is generated. The ADC Commands List Control Register CLCR1/2 defines the list entry point that is the first command to be executed. Subsequent list pointers are generated in an incremental way as the commands in the list are executed. When a trigger is linked to an eTimer or an EXT_TRG shown in the above figure, an event is generated on the corresponding output. The external trigger output has two modes of operation configured by TGSCR register bit ET_M:

- Pulse mode: a pulse with one system clock width is generated at the output pin
- Toggle mode: the output pin toggles its state as a result of each trigger event

An *ON-Time* and *Guard-Time* are defined for both Pulse and Toggle modes. The COTR register, Control *On-Time* Register, defines the (*ON-Time + Guard-Time)* duration. After a trigger has occurred a new trigger will be generated only after (*ON-time + Guard-Time)*. The COTR register only applies to external triggers; it does not affect ADC commands.

40.7.6 ADC commands list

The SU implements a command list that stores the ADC commands. The command list stores up to 24 x 16-bit commands (see ADC command list format) in a double-buffered implementation. The command list buffer registers may be updated at any time between two consecutive MRS, but the changes are transferred from the buffers to the actual registers only after an MRS occurs, so a correct reload procedure is performed (see Reload mechanism). The first command in a list is pointed by the 5-bit CLCRx register. Once a command list is triggered, it executes until the last command is found. The last command (n) is defined by the LC bit of the next command (n+1) in the list. Note that the command with LC=1 does not belong to the list and thus it is not executed. Note also that the LC bit is ignored in the first command of a list. See the command list registers, CLR_A_n, CLR_B_n, or CLR_C_n, where n = 1,….,24. The figure below describes the state diagram for the Commands List execution. A sequence of commands on the list is executed until the last command is found. See ADC command list format for more details.

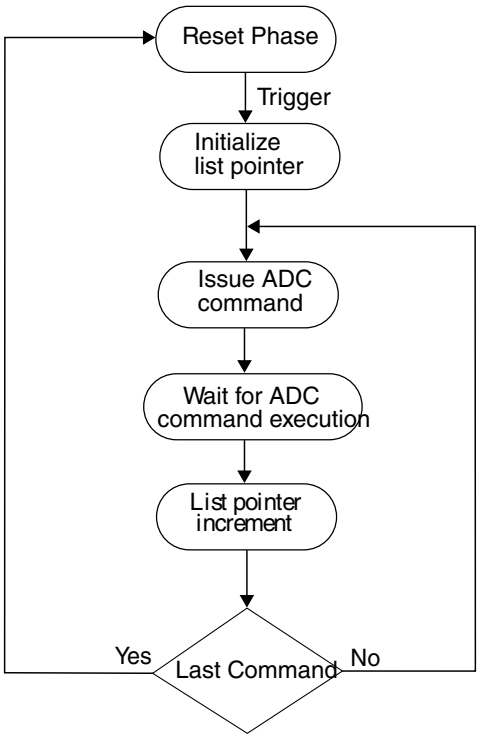


Figure 40-8. ADC command list scanning state diagram

Note

The CTU reads the next command line (the LC bit) to determine if the present command is the last one to be executed.

40.7.7 ADC command list format

The ADC command can be configured for single or dual conversion. In single conversion mode, the command targets one ADC only. In dual conversion mode, the conversion command is sent to both ADCs at the same time. Due to the clock domain crossing between CTU and ADCs, there could be a misalignment of up to one ADC clock cycle between the two commands.

The result of each conversion, in both modes, is stored in one of the four available FIFOs. In dual conversion mode, both ADCs store the result of their conversion in the same FIFO. If both ADCs access the FIFO in the same clock cycle, ADC unit A has higher priority, otherwise the first available conversion result is stored first.

The CTU supports up to 32 channels for both ADCs. The same physical channel cannot be assigned for both ADCs in a dual conversion mode. If there is an attempt to do so, an error is flagged in the EFR register's ICE bit, but the CTU sends the command to both ADCs.

In general, an ADC command is composed by the following fields depending upon the conversion format used (see Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_*n*)):

- Channel A: ADC A channel number (4 bits)
- Channel B: ADC B channel number (4 bits)
- Target ADC selection: ADC A or ADC B used in single conversion mode only (1 bit)
- FIFO selection bits for the selected ADC unit: up to four FIFOs (2 bits)
- Conversion mode selection: single or dual conversion mode (1 bit)
- Last command (LC): defines the last command in a list (1 bit)
- Interrupt request: enable interrupt request on command execution (1 bit)

The first command of a list is indicated by the CLCR1/2 registers' Tx_INDEX field. The last command is indicated by the LC (Last Command bit) of the following command in the command list. The first command is always executed. The execution of the following commands is controlled by the LC bit.

The figure below describes several scenarios for the command sequencing. If the first command indicated by the CLCR1/2 register is one of the lines identified as "Last command," then the list is a single command list. Note also that the list pointer wraps back to the first command in the list after command 24 is executed and if it is not the last command. If no command is defined as the last command, then the list will be executed continuously in a loop of 24 commands.

- Command List A: two command queues are available with elements 4 and 24 being last commands
- Command List B: sub-list 1 with one single command at address 1, and sub-list 2 with a queue of commands with last command at address 24
- Command List C: sub-list 2 with one single command at address 24, and sub-list 1 with a command queue with last command at address 23
- Command List D: two command queues with a wrap at address 24

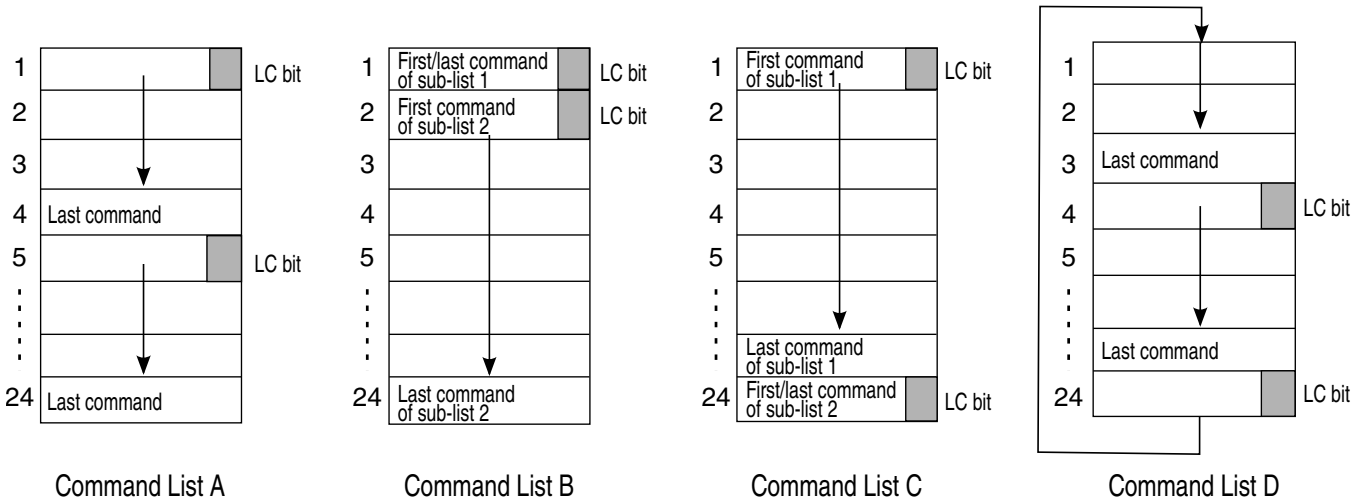


Figure 40-9. Command list sequencing

Note

The addresses shown in in the above figure do not represent addresses in the memory map. Please refer to the Memory Map section to check for the actual addresses.

40.7.8 ADC command list operation modes

There are two modes of operation regarding the ADC command list execution: ***streaming*** mode and ***parallel*** mode. In streaming mode, the command lists should behave as a stream of commands, meaning that two or more lists cannot be executed at the same time, but only in a sequence. Thus if list 1 is triggered, it should finish the execution of its last command before list 2 can be triggered. If there is an attempt to trigger more than one list at the same time, then an error occurs.

In parallel mode, up to two lists can be executed at the same time. In order to avoid errors during the execution of two parallel lists, they should not have commands for both ADCs. Therefore, if list 1 has commands for ADC A, then list 2 should only have commands for ADC B. Note also that if a shared analog input is used by one of the lists, it should not be used by the other list in order to avoid a contention and therefore an error. ADC dual conversion commands should also not be used in parallel mode.

Figure 40-10 shows the execution of two lists in streaming mode. While a list is being executed, any attempt to trigger the execution of another list while in this mode will cause an error. Note that both lists may have commands for ADC A or B with no restriction. Also, dual commands may be used.

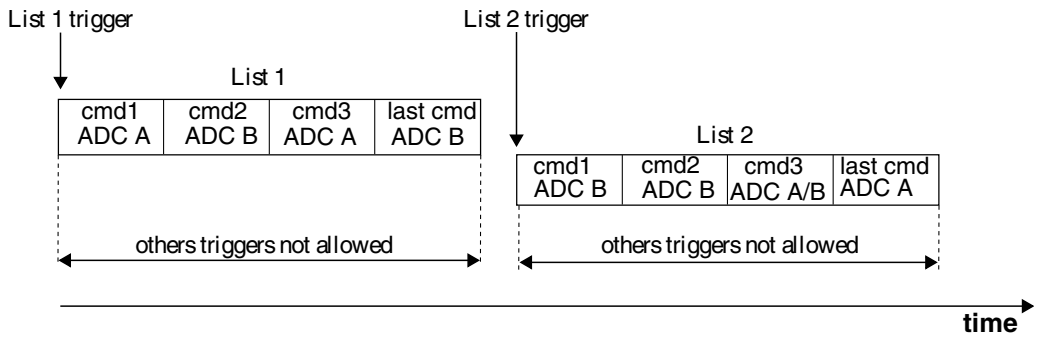


Figure 40-10. List execution in Streaming Mode

Figure 40-11 shows the execution of two lists in parallel mode. There are no restrictions for the triggers of both lists. After the two lists are triggered, no other triggers are allowed. If there is an attempt to trigger a third list while lists 1 and 2 are executing, then an error will be issued by setting ADC_OE bit in EFR register. Note that list 1 only has commands for ADC A and list 2 only has commands for ADC B. Having ADC A and ADC B commands in the same list may cause an error since two commands for the same ADC could be executed from different lists at the same time.

Note

The mixing of commands for both ADCs in the same list in parallel mode must not be used since the correct execution of commands in this case is not guaranteed by the CTU hardware.

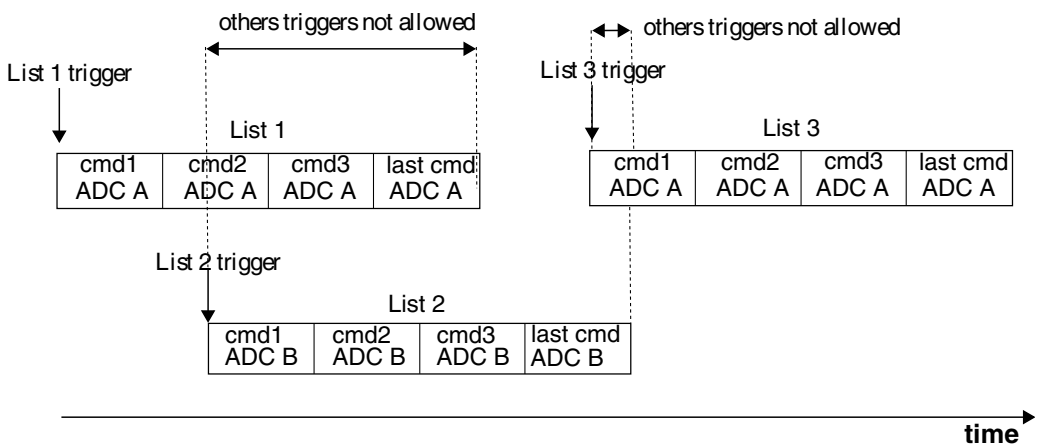


Figure 40-11. List execution in parallel mode

The list mode is controlled by the PAR_LIST bit in the LISTCSR register.

40.7.9 ADC results FIFOs

ADC results are stored in one of the four FIFOs. The FIFOs allow the storage of ADC results according to the type of data acquisition (phase currents, rotor position, ground-noise, and so on) so the software can distinguish data from several ADC channels. Each FIFO has its own interrupt line, DMA request signal, and a status register. The target FIFO for a conversion result is specified in the ADC command. Each FIFO element is 32 bits wide. The following figure shows the FIFO selection for ADC A and ADC B results.

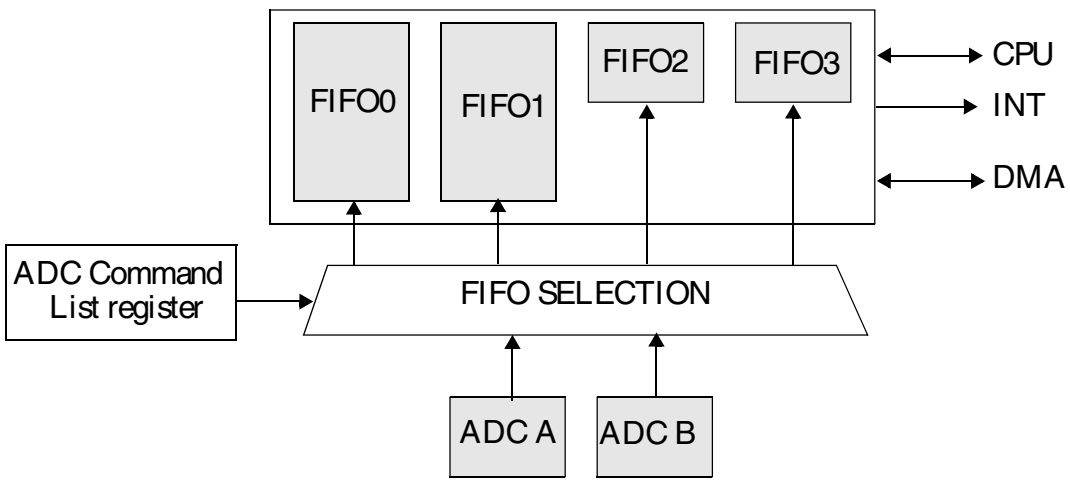


Figure 40-12. FIFO selection for ADC results storage

The CTU FIFOs are fixed depth. For example:

- FIFO0 and FIFO1: 16 entries each, dimensioned for full PWM period current acquisitions
- FIFO2 and FIFO3: 4 entries each, dimensioned for low rate acquisitions

FIFO access can be done by 16-bit or 32-bit read transaction. In 32-bit reads the conversion result, the target ADC, and the channel number are accessed. 8-bit access must not be used; otherwise, the FIFO pointer increments and the complete data is not retrieved. The FIFO result register can be read in a right- or left-aligned format using two different addresses:

- Unsigned right-justified, read from register FLx (for FIFO x)

  (Conversion result is unsigned right-justified data, i.e. bits [11:0] are used for 12-bit resolution and bits [15:12] always return zero)
- Signed left-justified, read from register FLx (for FIFO x)

(Conversion result is signed left-justified data, i.e. bit [15] is reserved for sign and is always read as zero, bits [14:3] are used for 12-bit resolution and bits [2:0] always return zero)

The FIFO access should be based on the status bits in the FCR register. Empty and full indication are provided. If data is read from an empty FIFO, then the read data should not be considered as valid data. Note that there is no indication of an underflow condition. If the FIFO is full and there is an attempt to write more data, then the new data is not written to the FIFO and an overrun flag is set.

40.7.10 Reload mechanism

In order to assure coherent programming, several CTU registers are double-buffered, thus allowing the programming of new data on the buffers while not affecting data being used. All registers are uploaded with buffer data at the same time. For the majority of the CTU registers the re-load is controlled by the Master Reload Signal, MRS.

Since the Master Reload Signal is generated by the hardware, it may occur while the software is still updating the buffer registers. In this case, noncoherent values are written to the registers because the CPU did not finish the programing of all registers. In order to avoid this situation, a General Reload Enable, GRE, control bit is provided. If GRE is cleared then no reload occurs. If this bit is set, then the reload is done when MRS occurs. The GRE bit is automatically cleared by MRS or is cleared by software using the CGRE bit in the same register. The CGRE bit is self-cleared and thus always read as zero.

The software should be able to program all registers buffers within a control cycle. In order to help the software to evaluate that a flawless register programing was executed, the MRS_RE flag is provided in the EFR register. If cleared, this flag indicates that all register buffers were programed before MRS occurred, as follows:

- If any of the double buffered registers is written, the FGRE flag is set indicating that a programing cycle has initiated.
- Set GRE to indicate that all updates to the buffer registers were performed.
- The reload of all registers is executed when the MRS occurs. Since GRE is set then MRS_RE will not be set.
- GRE and FGRE are cleared when MRS occurs.

See Figure 40-13 for more details.

An error is flagged by MRS_RE when MRS occurs, GRE is cleared, and FGRE is set. This scenario indicates that the software has initiated a register programming but has not finished it before the MRS, as follows:

- If any of the double buffered registers is written, the FGRE flag is set indicating that a programing cycle has initiated.
- MRS occurs and GRE is not set. In this case MRS_RE is set and, if enabled, an interrupt is generated.
- FGRE is not cleared when MRS occurs and no register is updated, since GRE is cleared.

Note that the software does not need to update all double-buffered registers during a *control cycle*. In order to avoid the error indication it is sufficient to set GRE before a MRS occurs.

A race condition occurs when GRE is being cleared by CGRE at the same time that an MRS occurs. In this case, GRE is considered to be cleared and no reload occurs. Another race condition may occur between CGRE and GRE being set. Since these two bits are in the same register they could eventually be set at the same time. In this case, CGRE has precedence; thus, GRE is cleared.

Note

MRS has priority over the TGSISR_RE bit in the CR register, which is the re-synchronization bit of the TGSISR.

The following figure describes two scenarios for the register re-load. Scenario (a) is a normal re-load where the buffers of the double-buffered registers are programmed and the GRE bit is set before the next MRS occurs. The scenario presented in (b) describes an error condition where the software did not signal the hardware that all registers are ready for re-load before the next MRS. This is indicated by GRE being at zero when MRS occurs. In this case, MRS_RE flag is set indicating the fault condition.

Figure 40-13. Reload scenarios

40.8 Interrupts and DMA Requests

40.8.1 DMA support

The DMA interface can be used to configure the CTU registers. A DMA request is issued if MRS occurs and the GRE bit is set. In this case, the previously transferred data is loaded from the buffers into the registers and a new transfer cycle is initiated with the DMA request. The DMA done signal from the DMA controller is used to set GRE bit which allows the reload to be performed. The following figure describes this operation.

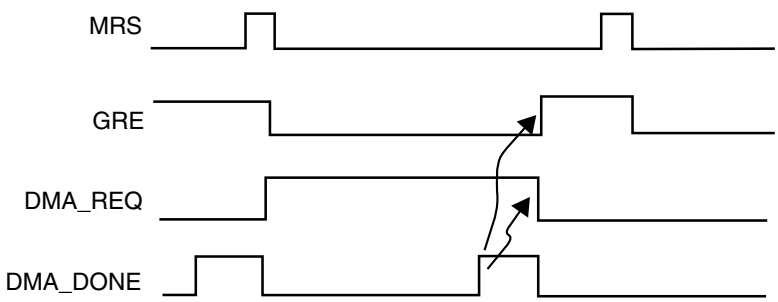


Figure 40-14. Register reload using DMA

DMA support can also be provided for reading FIFO stored data. Each FIFO can be configured to perform a DMA request when the number of stored words reaches a threshold value defined in the FTH register. See the FIFO Threshold, FTH, register for more information. After a DMA_DONE and the remaining data in the FIFO is below the watermark, then the DMA request is removed. Thus for an efficient operation, the DMA should be configured to execute a loop reading all data in the FIFO considering the amount of data the same as defined by the watermark.

40.8.2 CTU faults and errors

The CTU detects and signals the following error conditions:

- MRS reload error: MRS occurs while user is updating the double-buffered registers.
- Sequential mode trigger overrun SM_TO: more than 8 event signals occur before the next MRS in TGS sequential mode.

- Trigger event overrun indicated by 4 interrupt signals ADC_OE, T0_OE, T1_OE and ET_OE: a trigger event occurs during the time frame when the previous trigger was not completed.
- The CTU allows the user to pre-set a trigger to the eTimer1 in the middle of an ADC conversion. In parallel mode even two ADCs' commands can be issued at the same time. The error only occurs if the second trigger is targeting the same ADC.
- Invalid (unrecognized) ADC command occurs and the ICE bit is set.
- Master reload signal overrun MRS_O: MRS occurs before all triggers selected by (Tn_E and (TnETE or TnTmE or Tn_ADCE)) bits in THCR1/2 registers have occurred.
- TGS_OSM TGS overrun in sequential mode: a new event occurs before the trigger event selected by the previous event had occurred.

The faults/errors flags in the CTU error flag register and in the CTU interrupt flag register can be cleared by writing 1 to the respective bit. Writing 0 to these bits has no effect.

Note

The CTU does not support write-protection mechanism for any register, meaning that all registers are accessible at any time.

The CTU does not generate a transfer error when its reserved address space is accessed.

40.8.2.1 CTU parallel list errors

The command list operation in parallel mode may generate errors since two lists may be in execution at the same time. Below some error conditions are described.

Commands for the same ADC are issued at the same time by two lists. In this case:

- One of the commands is disregarded and an error is issued.
- The LIST_BE bit in the EFR register is set.
- The address of the list commands are shown in the LISTCSR register.
- The address of the command that was not executed is indicated in the LISTCSR register.

A third list is triggered when two lists are already being executed. In this case:

- The third list is not executed and an error is indicated by the ADC_OE bit in the EFR register.

One list is being executed and a second list is triggered; the first command of the second list is for the same ADC being used by the first list. In this case:

- The command of the second list is not executed.
- The address of the second list command is shown in the LISTCSR register.
- The LIST_BE bit in the EFR register is set.

40.8.3 CTU interrupt/DMA requests

The CTU generates the following interrupt/DMA requests which are controlled by the IR register:

- Error interrupt request, see CTU faults and errors (1 interrupt line)
- ADC command interrupt request (1 interrupt line).
- MRS interrupt request (1 interrupt line)
- Trigger event interrupt request (1 interrupt line for each trigger event)
- FIFOs interrupt and/or DMA transfer request (1 interrupt line for each FIFO). See FIFO control register, FCR.
- DMA transfer request on the MRS occurrence if GRE bit is set

40.9 Self test mode

The CTU is capable of issuing a self test command to the ADC.

The CLRx register is used to set up a self test command. Table 40-4 describes the ST0 and ST1 bits that control self-test functionality.

Table 40-4. Self-test command

ST_CMS bit	ST1 bit	ST0 bit	Command description
0	Not Used	0	no self test single conversion
1	Not Used	0	no self test dual conversion
Not Used	0	1	self test command

Table continues on the next page...

Table 40-4. Self-test command (continued)

ST_CMS bit	ST1 bit	ST0 bit	Command description
0	1	1	no self test command single conversion
1	1	1	no self test command dual conversion

Two algorithms are available in self test mode, see Table 40-5.

Table 40-5. Self-test algorithm selection

ALG[1]	Number of steps	Executed Algorithm
00	3	Algorithm S
01[2]	–	Reserved
10	12	Algorithm C
11	15	Algorithm FULL (S + C)

1. ALG refers to the CLRx register field.
2. The 01 selection causes a command execution error and the LIST_BE flag is set in the Error Flag Register (EFR).

The self-test S algorithm is executed in an atomic burst of conversions independent of the BSIZE value in the CLRx register. Below are described the steps for this algorithm execution.

- Step 0: VBGAP/VREF test
- Step 1: VDD/VREF test
- Step 2: VREF/VREF test

When the FULL algorithm is selected, the CTU executes the S algorithm followed by the C algorithm.

The self-test execution follows the trigger scheme selected for the CTU, either triggered mode or sequential mode can be selected.

In order to completely execute one of the algorithms described in Table 40-5 a certain number of steps need to be performed. Each step corresponds to the time for one ADC conversion to complete. These steps may be executed at once, by just one command, or they can be interleaved with other ADC commands. In order to executed self-test commands interleaved with other conversion commands it is sufficient to set the BSIZE field in the CLRx register for a value lower than the number of steps defined for the selected algorithm. The following figure shows the execution of the C algorithm interleaved with regular ADC commands.

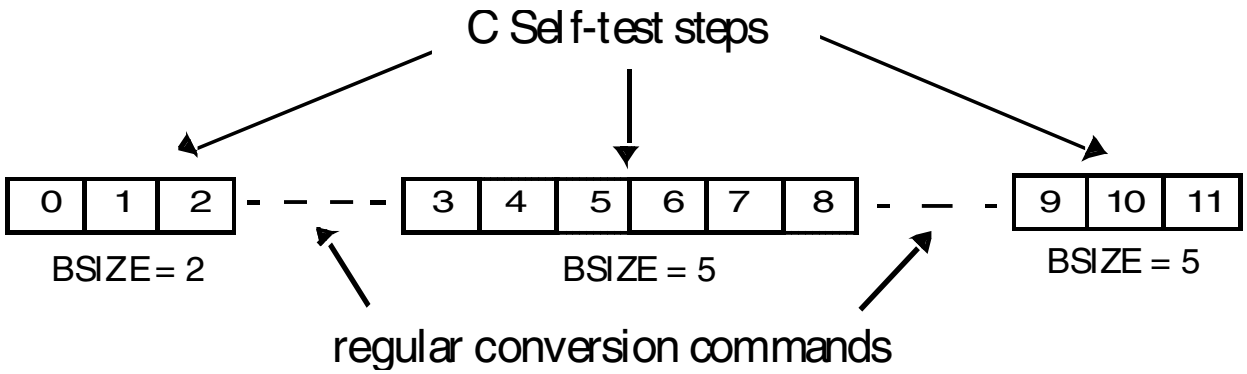


Figure 40-15. C Algorithm execution interleaved with others ADC commands

In the figure above, the last C self-test sequence intentionally shows BSIZE set to a larger value than the remaining steps needed to complete the C algorithm. In this case, the number of executed self-test steps does not follow BSIZE but instead it follows the number of remaining steps for the selected self-test algorithm, which in this case is 3 steps.

Note

The S algorithm steps do not interleave with other ADC commands even if BSIZE is smaller than 3. Independent of BSIZE value all the 3 steps of the S algorithm are executed in a contiguous sequence.

The number of steps needed to completely execute the selected self-test algorithm is as follows:

- S algorithm requires 3 steps
- C algorithm requires 12 steps
- FULL runs all algorithm and requires 15 steps

In the figure below, the FULL algorithm is executed interleaved with other ADC commands. The BSIZE = 3 in the first and second self-test commands produces different number of executed self-test steps.

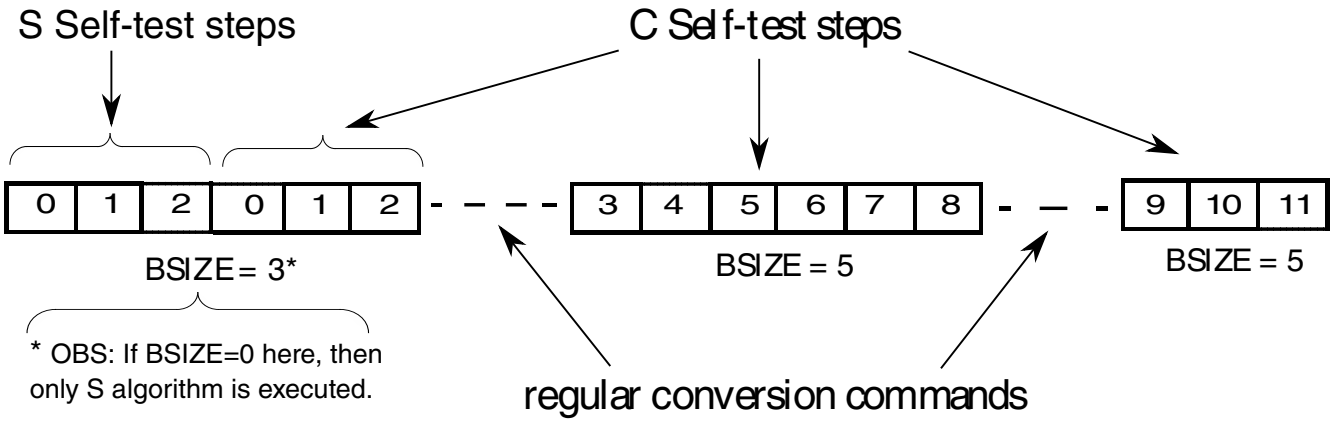


Figure 40-16. FULL algorithm execution interleaved with other ADC commands

40.9.1 Self-test execution errors

When a self-test algorithm is initiated for an ADC, it should complete before any other self-test algorithm is started in the same ADC. In other words, one self-test algorithm should not be interleaved with another self-test algorithm. Any attempt to do so will cause the ongoing list execution to be halted and cause the LIST_BE error flag (see LIST_BE) to be set. In order to evaluate if a self-test algorithm is running, the CS bit (see CS) can be checked. If set, this bit indicates a self-test has initiated in one of the ADCs. If a self-test algorithm conflict occurs, as described, the recovery procedure is to issue a software reset through the CTU_ADC_R bit in the CR register.

An error condition (see LIST_BE) is also caused when the reserved value 01 is selected for the self-test algorithm.

Chapter 41
Enhanced Motor Control Timer (eTimer)

41.1 Chip-specific eTimer information

41.1.1 Enhanced motor control timer (eTimer) configuration

This device contains two eTimer modules: eTIMER_1 and eTIMER_2.

The following table lists the eTimer primary timer/counter channel connections.

Table 41-1. eTimer primary timer/counter channel connections

Module	Channel	Source/Target
eTIMER_1	CH0-5 (Input)	From PAD (Refer to attached IO Signal Description and Input multiplexing tables spreadsheet)
eTIMER_1	CH6-7 (Input)	Reserved
eTIMER_2	CH0-5 (Input)	From PAD (Refer to attached IO Signal Description and Input multiplexing tables spreadsheet)
eTIMER_2	CH6-7 (Input)	Reserved
eTIMER_1	CH0-5 (Output)	To PAD (Refer to attached IO Signal Description and Input multiplexing tables spreadsheet)
eTIMER_1	CH6 (Output)	Reserved
eTIMER_1	CH7 (Output)	ENET - 1588 TIMER Channel0
eTIMER_2	CH0-5 (Output)	To PAD (Refer to attached IO Signal Description and Input multiplexing tables spreadsheet)
eTIMER_2	CH6-7 (Output)	Reserved

NOTE

Not all eTimer channels are available on the chip pads. Certain channels are connected internally with the ENET module.

NOTE

Interrupt for Channel7 of eTIMER_1 is disabled.

Table 41-2. eTimer module configurations

Module	Total number of channels	Auxiliary inputs	DMA channels	Channel used for injection conversion support
eTIMER_1	8	7	2	5
eTIMER_2	8	7	2	

The following table lists the eTIMER auxiliary connections.

Table 41-3. eTimer auxiliary connections

Input module	Input signal	Driving module	Driver signal
eTIMER_1	AUX_0	CTU_0	eTIMER1_TRG
eTIMER_1	AUX_1	eTIMER_2	T2
eTIMER_1	AUX_2	FlexRay	FR_CA_TX
eTIMER_1	AUX_3	SPI_1	SCK
eTIMER_1	AUX_4	CTE	AUX_0
eTIMER_1	AUX_5	MIPICSI2	SOP
eTIMER_1	AUX_6	ENET	TIMER_3
eTIMER_1	AUX_7	Reserved	-
eTIMER_2	AUX_0	Reserved	-
eTIMER_2	AUX_1	Reserved	-
eTIMER_2	AUX_2	CTU_0	eTIMER2_TRG
eTIMER_2	AUX_3	eTIMER_1	T4
eTIMER_2	AUX_4	MIPICSI2	EOP
eTIMER_2	AUX_5	CTE	AUX_1
eTIMER_2	AUX_6	ENET	TIMER_3
eTIMER_2	AUX_7	Reserved	-

NOTE

In above table, eTIMER1_TRG and eTIMER2_TRG triggers corresponds to ETIMER2_TRG and ETIMER3_TRG triggers in the Figure 40-1.

41.1.1.1 eTimer register accesses

NOTE

Address range from 10Ch to 116h will not give any transfer error for unimplemented registers in between in this range and writing on unimplemented spaces in between this range is not advised.

41.2 Introduction

This section briefly describes the eTimer, its features and customization, and contains block diagrams.

41.2.1 Overview

An eTimer module provides:

- Eight identical counter/timer channels

Each 16-bit counter/timer channel contains a prescaler, a counter, a load register, a hold register, two queued capture registers, two compare registers, two compare preload registers, and four control registers.

Note

This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter's terminal value has been reached. For true modulo counting the counter can also be initialized by the CMPLD1 or CMPLD2 registers.

The Hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters coherently.

The Capture registers enable an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The Prescaler provides different time bases useful for clocking the counter/timer.

The Counter provides the ability to count internal or external events.

Within the eTimer module, the input pins are shareable.

41.2.2 Features

The eTimer module design includes these distinctive features:

- 16-bit counters/timers: Eight.
- Count up/down.
- Counters are cascadable.
- Enhanced programmable up/down modulo counting.
- Max count rate equals peripheral clock/2 for external clocks.
- Max count rate equals peripheral clock for internal clocks.
- Count once or repeatedly.
- Counters are preloadable.
- Compare registers are preloadable.
- Counters can share available input pins.
- Separate prescaler for each counter.
- Each counter has capture and compare capability.
- Continuous and single shot capture for enhanced speed measurement.
- DMA support of capture registers and compare registers.
- OFLAG comparison for safety critical applications.
- Programmable operation during debug mode .
- Programmable input filter.
- Counting start can be synchronized across counters.

41.2.3 Module block diagram

The eTimer block diagram is shown in the following figure.

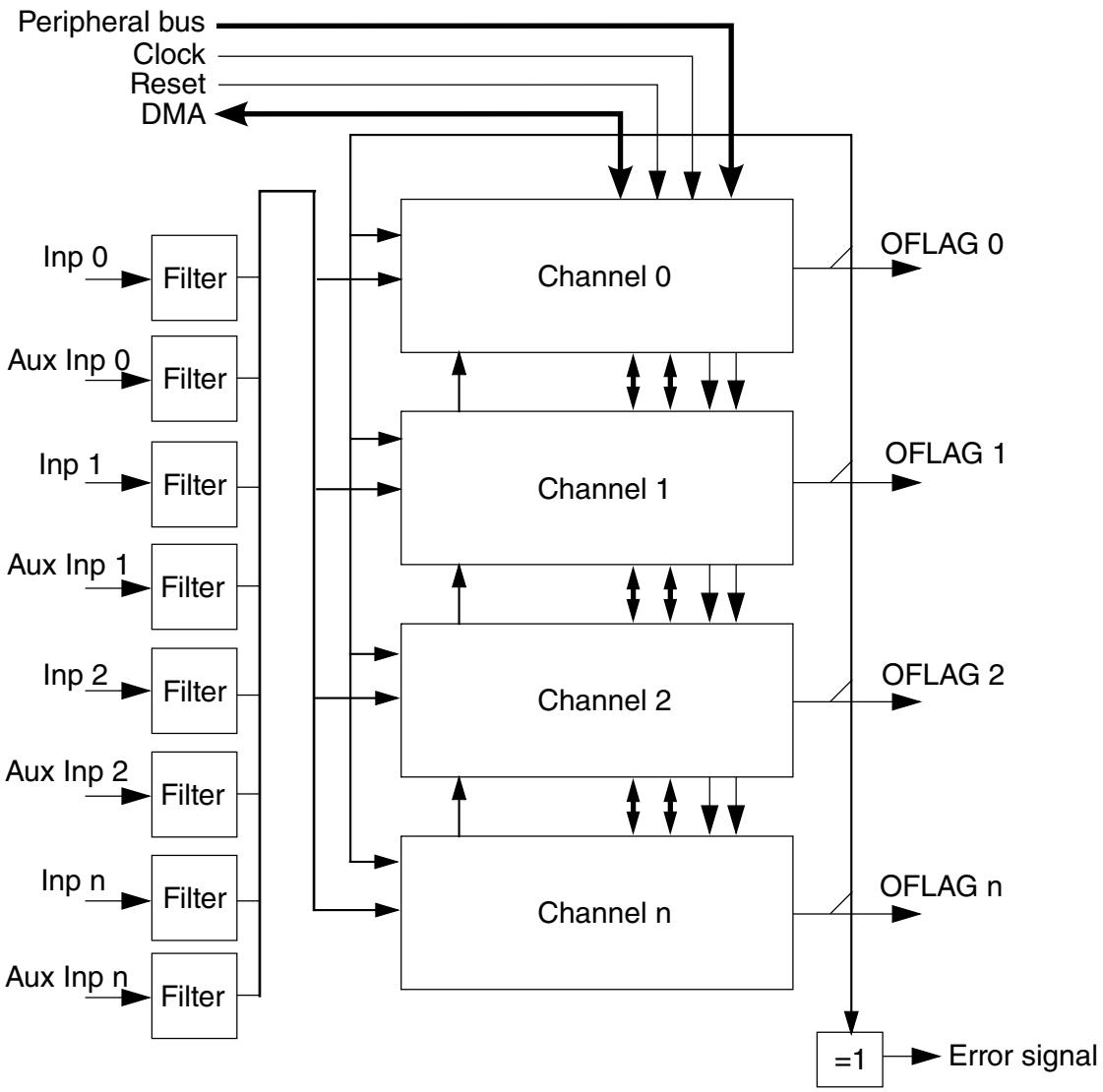


Figure 41-1. eTimer Block Diagram

41.2.4 Channel Block Diagram

Each of the timer/counter channels within the eTimer are shown in the following figure.

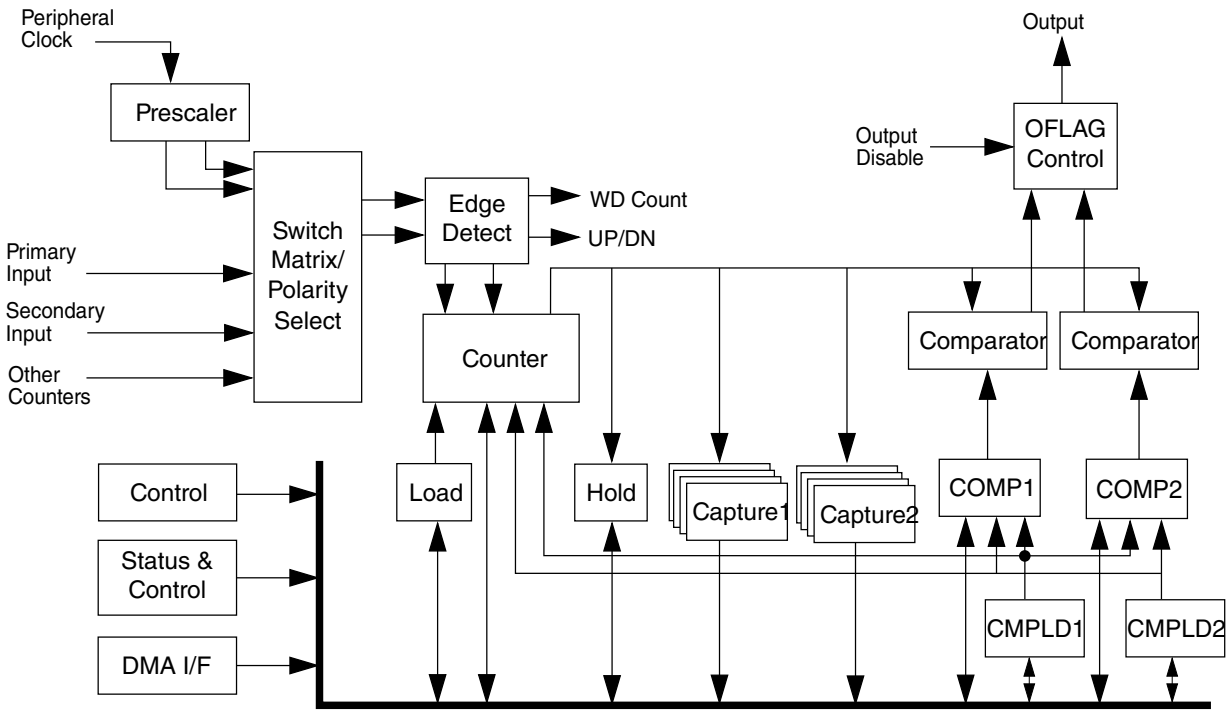


Figure 41-2. eTimer Channel Block Diagram

41.3 External Signal Descriptions

Each eTimer module has eight external signals that can be used as either inputs or outputs. The number of auxiliary inputs per module is device-dependent. See the configuration section for more information. The eTimer also interfaces to the Peripheral Bus.

41.3.1 TIO[n:0] - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags. In Figure 41-2, the TIO signals are shown as the Primary Inputs, and the Outputs from the OFLAG Control block.

41.3.2 TAI[n:0] - Timer Auxiliary Inputs

These pins act as alternate input choices for the timer channels. The TAI signals are shown as Aux Inp 1..n in Figure 41-1.

41.4 Memory map and register definition

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level.

There are a set of registers for each timer channel, and a set of configuration registers.

Certain registers are not byte-accessible. Registers which generate an error when a byte access occurs are noted as such in their register descriptions. Other registers which are not byte-accessible will not generate an error when a byte access occurs.

A write to a read-only register will produce an error.

ETIMER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel n Compare Register 1 (ETIMER_CH0_COMP1)	16	R/W	0000h	41.4.1/1530
2	Channel n Compare Register 2 (ETIMER_CH0_COMP2)	16	R/W	0000h	41.4.2/1530
4	Channel n Capture Register 1 (ETIMER_CH0_CAPT1)	16	R	0000h	41.4.3/1531
6	Channel n Capture Register 2 (ETIMER_CH0_CAPT2)	16	R	0000h	41.4.4/1531
8	Channel n Load Register (ETIMER_CH0_LOAD)	16	R/W	0000h	41.4.5/1532
A	Channel n Hold Register (ETIMER_CH0_HOLD)	16	R	0000h	41.4.6/1532
C	Channel n Counter Register (ETIMER_CH0_CNTR)	16	R/W	0000h	41.4.7/1533
E	Channel n Control Register 1 (ETIMER_CH0_CTRL1)	16	R/W	0000h	41.4.8/1533
10	Channel n Control Register 2 (ETIMER_CH0_CTRL2)	16	R/W	0000h	41.4.9/1536
12	Channel n Control Register 3 (ETIMER_CH0_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
14	Channel n Status Register (ETIMER_CH0_STS)	16	R/W	0000h	41.4.11/ 1539
16	Channel n Interrupt and DMA Enable Register (ETIMER_CH0_INTDMA)	16	R/W	0000h	41.4.12/ 1540
18	Channel n Comparator Load Register 1 (ETIMER_CH0_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
1A	Channel n Comparator Load Register 2 (ETIMER_CH0_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
1C	Channel n Compare and Capture Control Register (ETIMER_CH0_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
1E	Channel n Input Filter Register (ETIMER_CH0_FILT)	16	R/W	0000h	41.4.16/ 1545
20	Channel n Compare Register 1 (ETIMER_CH1_COMP1)	16	R/W	0000h	41.4.1/1530
22	Channel n Compare Register 2 (ETIMER_CH1_COMP2)	16	R/W	0000h	41.4.2/1530
24	Channel n Capture Register 1 (ETIMER_CH1_CAPT1)	16	R	0000h	41.4.3/1531

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
26	Channel n Capture Register 2 (ETIMER_CH1_CAPT2)	16	R	0000h	41.4.4/1531
28	Channel n Load Register (ETIMER_CH1_LOAD)	16	R/W	0000h	41.4.5/1532
2A	Channel n Hold Register (ETIMER_CH1_HOLD)	16	R	0000h	41.4.6/1532
2C	Channel n Counter Register (ETIMER_CH1_CNTR)	16	R/W	0000h	41.4.7/1533
2E	Channel n Control Register 1 (ETIMER_CH1_CTRL1)	16	R/W	0000h	41.4.8/1533
30	Channel n Control Register 2 (ETIMER_CH1_CTRL2)	16	R/W	0000h	41.4.9/1536
32	Channel n Control Register 3 (ETIMER_CH1_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
34	Channel n Status Register (ETIMER_CH1_STS)	16	R/W	0000h	41.4.11/ 1539
36	Channel n Interrupt and DMA Enable Register (ETIMER_CH1_INTDMA)	16	R/W	0000h	41.4.12/ 1540
38	Channel n Comparator Load Register 1 (ETIMER_CH1_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
3A	Channel n Comparator Load Register 2 (ETIMER_CH1_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
3C	Channel n Compare and Capture Control Register (ETIMER_CH1_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
3E	Channel n Input Filter Register (ETIMER_CH1_FILT)	16	R/W	0000h	41.4.16/ 1545
40	Channel n Compare Register 1 (ETIMER_CH2_COMP1)	16	R/W	0000h	41.4.1/1530
42	Channel n Compare Register 2 (ETIMER_CH2_COMP2)	16	R/W	0000h	41.4.2/1530
44	Channel n Capture Register 1 (ETIMER_CH2_CAPT1)	16	R	0000h	41.4.3/1531
46	Channel n Capture Register 2 (ETIMER_CH2_CAPT2)	16	R	0000h	41.4.4/1531
48	Channel n Load Register (ETIMER_CH2_LOAD)	16	R/W	0000h	41.4.5/1532
4A	Channel n Hold Register (ETIMER_CH2_HOLD)	16	R	0000h	41.4.6/1532
4C	Channel n Counter Register (ETIMER_CH2_CNTR)	16	R/W	0000h	41.4.7/1533
4E	Channel n Control Register 1 (ETIMER_CH2_CTRL1)	16	R/W	0000h	41.4.8/1533
50	Channel n Control Register 2 (ETIMER_CH2_CTRL2)	16	R/W	0000h	41.4.9/1536
52	Channel n Control Register 3 (ETIMER_CH2_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
54	Channel n Status Register (ETIMER_CH2_STS)	16	R/W	0000h	41.4.11/ 1539
56	Channel n Interrupt and DMA Enable Register (ETIMER_CH2_INTDMA)	16	R/W	0000h	41.4.12/ 1540
58	Channel n Comparator Load Register 1 (ETIMER_CH2_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
5A	Channel n Comparator Load Register 2 (ETIMER_CH2_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
5C	Channel n Compare and Capture Control Register (ETIMER_CH2_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
5E	Channel n Input Filter Register (ETIMER_CH2_FILT)	16	R/W	0000h	41.4.16/ 1545

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	Channel n Compare Register 1 (ETIMER_CH3_COMP1)	16	R/W	0000h	41.4.1/1530
62	Channel n Compare Register 2 (ETIMER_CH3_COMP2)	16	R/W	0000h	41.4.2/1530
64	Channel n Capture Register 1 (ETIMER_CH3_CAPT1)	16	R	0000h	41.4.3/1531
66	Channel n Capture Register 2 (ETIMER_CH3_CAPT2)	16	R	0000h	41.4.4/1531
68	Channel n Load Register (ETIMER_CH3_LOAD)	16	R/W	0000h	41.4.5/1532
6A	Channel n Hold Register (ETIMER_CH3_HOLD)	16	R	0000h	41.4.6/1532
6C	Channel n Counter Register (ETIMER_CH3_CNTR)	16	R/W	0000h	41.4.7/1533
6E	Channel n Control Register 1 (ETIMER_CH3_CTRL1)	16	R/W	0000h	41.4.8/1533
70	Channel n Control Register 2 (ETIMER_CH3_CTRL2)	16	R/W	0000h	41.4.9/1536
72	Channel n Control Register 3 (ETIMER_CH3_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
74	Channel n Status Register (ETIMER_CH3_STS)	16	R/W	0000h	41.4.11/ 1539
76	Channel n Interrupt and DMA Enable Register (ETIMER_CH3_INTDMA)	16	R/W	0000h	41.4.12/ 1540
78	Channel n Comparator Load Register 1 (ETIMER_CH3_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
7A	Channel n Comparator Load Register 2 (ETIMER_CH3_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
7C	Channel n Compare and Capture Control Register (ETIMER_CH3_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
7E	Channel n Input Filter Register (ETIMER_CH3_FILT)	16	R/W	0000h	41.4.16/ 1545
80	Channel n Compare Register 1 (ETIMER_CH4_COMP1)	16	R/W	0000h	41.4.1/1530
82	Channel n Compare Register 2 (ETIMER_CH4_COMP2)	16	R/W	0000h	41.4.2/1530
84	Channel n Capture Register 1 (ETIMER_CH4_CAPT1)	16	R	0000h	41.4.3/1531
86	Channel n Capture Register 2 (ETIMER_CH4_CAPT2)	16	R	0000h	41.4.4/1531
88	Channel n Load Register (ETIMER_CH4_LOAD)	16	R/W	0000h	41.4.5/1532
8A	Channel n Hold Register (ETIMER_CH4_HOLD)	16	R	0000h	41.4.6/1532
8C	Channel n Counter Register (ETIMER_CH4_CNTR)	16	R/W	0000h	41.4.7/1533
8E	Channel n Control Register 1 (ETIMER_CH4_CTRL1)	16	R/W	0000h	41.4.8/1533
90	Channel n Control Register 2 (ETIMER_CH4_CTRL2)	16	R/W	0000h	41.4.9/1536
92	Channel n Control Register 3 (ETIMER_CH4_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
94	Channel n Status Register (ETIMER_CH4_STS)	16	R/W	0000h	41.4.11/ 1539
96	Channel n Interrupt and DMA Enable Register (ETIMER_CH4_INTDMA)	16	R/W	0000h	41.4.12/ 1540
98	Channel n Comparator Load Register 1 (ETIMER_CH4_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
9A	Channel n Comparator Load Register 2 (ETIMER_CH4_CMPLD2)	16	R/W	0000h	41.4.14/ 1542

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C	Channel n Compare and Capture Control Register (ETIMER_CH4_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
9E	Channel n Input Filter Register (ETIMER_CH4_FILT)	16	R/W	0000h	41.4.16/ 1545
A0	Channel n Compare Register 1 (ETIMER_CH5_COMP1)	16	R/W	0000h	41.4.1/1530
A2	Channel n Compare Register 2 (ETIMER_CH5_COMP2)	16	R/W	0000h	41.4.2/1530
A4	Channel n Capture Register 1 (ETIMER_CH5_CAPT1)	16	R	0000h	41.4.3/1531
A6	Channel n Capture Register 2 (ETIMER_CH5_CAPT2)	16	R	0000h	41.4.4/1531
A8	Channel n Load Register (ETIMER_CH5_LOAD)	16	R/W	0000h	41.4.5/1532
AA	Channel n Hold Register (ETIMER_CH5_HOLD)	16	R	0000h	41.4.6/1532
AC	Channel n Counter Register (ETIMER_CH5_CNTR)	16	R/W	0000h	41.4.7/1533
AE	Channel n Control Register 1 (ETIMER_CH5_CTRL1)	16	R/W	0000h	41.4.8/1533
B0	Channel n Control Register 2 (ETIMER_CH5_CTRL2)	16	R/W	0000h	41.4.9/1536
B2	Channel n Control Register 3 (ETIMER_CH5_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
B4	Channel n Status Register (ETIMER_CH5_STS)	16	R/W	0000h	41.4.11/ 1539
B6	Channel n Interrupt and DMA Enable Register (ETIMER_CH5_INTDMA)	16	R/W	0000h	41.4.12/ 1540
B8	Channel n Comparator Load Register 1 (ETIMER_CH5_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
BA	Channel n Comparator Load Register 2 (ETIMER_CH5_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
BC	Channel n Compare and Capture Control Register (ETIMER_CH5_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
BE	Channel n Input Filter Register (ETIMER_CH5_FILT)	16	R/W	0000h	41.4.16/ 1545
C0	Channel n Compare Register 1 (ETIMER_CH6_COMP1)	16	R/W	0000h	41.4.1/1530
C2	Channel n Compare Register 2 (ETIMER_CH6_COMP2)	16	R/W	0000h	41.4.2/1530
C4	Channel n Capture Register 1 (ETIMER_CH6_CAPT1)	16	R	0000h	41.4.3/1531
C6	Channel n Capture Register 2 (ETIMER_CH6_CAPT2)	16	R	0000h	41.4.4/1531
C8	Channel n Load Register (ETIMER_CH6_LOAD)	16	R/W	0000h	41.4.5/1532
CA	Channel n Hold Register (ETIMER_CH6_HOLD)	16	R	0000h	41.4.6/1532
CC	Channel n Counter Register (ETIMER_CH6_CNTR)	16	R/W	0000h	41.4.7/1533
CE	Channel n Control Register 1 (ETIMER_CH6_CTRL1)	16	R/W	0000h	41.4.8/1533
D0	Channel n Control Register 2 (ETIMER_CH6_CTRL2)	16	R/W	0000h	41.4.9/1536
D2	Channel n Control Register 3 (ETIMER_CH6_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
D4	Channel n Status Register (ETIMER_CH6_STS)	16	R/W	0000h	41.4.11/ 1539
D6	Channel n Interrupt and DMA Enable Register (ETIMER_CH6_INTDMA)	16	R/W	0000h	41.4.12/ 1540

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D8	Channel n Comparator Load Register 1 (ETIMER_CH6_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
DA	Channel n Comparator Load Register 2 (ETIMER_CH6_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
DC	Channel n Compare and Capture Control Register (ETIMER_CH6_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
DE	Channel n Input Filter Register (ETIMER_CH6_FILT)	16	R/W	0000h	41.4.16/ 1545
E0	Channel n Compare Register 1 (ETIMER_CH7_COMP1)	16	R/W	0000h	41.4.1/1530
E2	Channel n Compare Register 2 (ETIMER_CH7_COMP2)	16	R/W	0000h	41.4.2/1530
E4	Channel n Capture Register 1 (ETIMER_CH7_CAPT1)	16	R	0000h	41.4.3/1531
E6	Channel n Capture Register 2 (ETIMER_CH7_CAPT2)	16	R	0000h	41.4.4/1531
E8	Channel n Load Register (ETIMER_CH7_LOAD)	16	R/W	0000h	41.4.5/1532
EA	Channel n Hold Register (ETIMER_CH7_HOLD)	16	R	0000h	41.4.6/1532
EC	Channel n Counter Register (ETIMER_CH7_CNTR)	16	R/W	0000h	41.4.7/1533
EE	Channel n Control Register 1 (ETIMER_CH7_CTRL1)	16	R/W	0000h	41.4.8/1533
F0	Channel n Control Register 2 (ETIMER_CH7_CTRL2)	16	R/W	0000h	41.4.9/1536
F2	Channel n Control Register 3 (ETIMER_CH7_CTRL3)	16	R/W	0F00h	41.4.10/ 1538
F4	Channel n Status Register (ETIMER_CH7_STS)	16	R/W	0000h	41.4.11/ 1539
F6	Channel n Interrupt and DMA Enable Register (ETIMER_CH7_INTDMA)	16	R/W	0000h	41.4.12/ 1540
F8	Channel n Comparator Load Register 1 (ETIMER_CH7_CMPLD1)	16	R/W	0000h	41.4.13/ 1542
FA	Channel n Comparator Load Register 2 (ETIMER_CH7_CMPLD2)	16	R/W	0000h	41.4.14/ 1542
FC	Channel n Compare and Capture Control Register (ETIMER_CH7_CCCTRL)	16	R/W	0000h	41.4.15/ 1543
FE	Channel n Input Filter Register (ETIMER_CH7_FILT)	16	R/W	0000h	41.4.16/ 1545
10C	Channel Enable Register (ETIMER_ENBL)	16	R/W	See section	41.4.17/ 1546
110	DMA Request 0 Select Register (ETIMER_DREQ0)	16	R/W	0000h	41.4.18/ 1546
112	DMA Request 1 Select Register (ETIMER_DREQ1)	16	R/W	0000h	41.4.19/ 1548

41.4.1 Channel n Compare Register 1 (ETIMER_CH*n*_COMP1)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP1 can be found in Usage of Compare Registers .

Address: 0h base + 0h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	COMP1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_COMP1 field descriptions

Field	Description
0–15 COMP1	Value used for comparison with the counter value

41.4.2 Channel n Compare Register 2 (ETIMER_CH*n*_COMP2)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP2 can be found in Usage of Compare Registers .

Address: 0h base + 2h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	COMP2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_COMP2 field descriptions

Field	Description
0–15 COMP2	Value used for comparison with the counter value

41.4.3 Channel n Capture Register 1 (ETIMER_CH*n*_CAPT1)

This read-only register stores the value captured from the counter. It is actually a 2-deep FIFO and not a single register. This register is not byte accessible. A byte access to this register will cause an error to be generated. CCCTRL[CPT1MODE] determines when a capture occurs.

Address: 0h base + 4h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPT1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CAPT1 field descriptions

Field	Description
0–15 CAPT1	Value captured from the counter

41.4.4 Channel n Capture Register 2 (ETIMER_CH*n*_CAPT2)

This read-only register stores the value captured from the counter. It is actually a 2-deep FIFO and not a single register. This register is not byte accessible. A byte access to this register will cause an error to be generated. CCCTRL[CPT2MODE] determines when a capture occurs.

Address: 0h base + 6h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPT2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CAPT2 field descriptions

Field	Description
0–15 CAPT2	Value captured from the counter

41.4.5 Channel n Load Register (ETIMER_CH*n*_LOAD)

This read/write register stores the value used to initialize the counter. This register is not byte accessible.

Address: 0h base + 8h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_LOAD field descriptions

Field	Description
0–15 LOAD	Value used to initialize the counter

41.4.6 Channel n Hold Register (ETIMER_CH*n*_HOLD)

This read-only register stores the counter's value whenever any of the counters within a module are read. This is used to support coherent reading of cascaded counters.

Address: 0h base + Ah offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	HOLD															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_HOLD field descriptions

Field	Description
0–15 HOLD	Stores the counter's value whenever any of the counters within a module are read.

41.4.7 Channel n Counter Register (ETIMER_CH*n*_CNTR)

This read/write register is the counter for this channel of the timer module. This register is not byte accessible.

Address: 0h base + Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CNTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CNTR field descriptions

Field	Description
0–15 CNTR	Counter for this channel of the timer module

41.4.8 Channel n Control Register 1 (ETIMER_CH*n*_CTRL1)

Address: 0h base + Eh offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read Write	CNTMODE			PRISRC				
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ONCE	LENGTH	DIR	SECSRC				
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL1 field descriptions

Field	Description
0–2 CNTMODE	Count mode These bits control the basic counting and behavior of the counter. 000 No Operation 001 Count rising edges of primary source (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1. If primary count source is IP bus clock, only rising edges are counted regardless of PIPS value.) 010 Count rising and falling edges of primary source (IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.) 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count primary source rising edges, secondary source specifies direction (1 = minus) (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1.)

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description
	110 Edge of secondary source triggers primary count till compare 111 Cascaded counter mode, up/down (Primary count source must be set to one of the counter outputs.)
3–7 PRISRC	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output as its primary count source is not a legal choice. The result is no counting. — 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Counter #6 input pin 00111 Counter #7 input pin 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Auxiliary input #2 pin 01011 Auxiliary input #3 pin 01100 Auxiliary input #4 pin 01101 Auxiliary input #5 pin 01110 Auxiliary input #6 pin 01111 Auxiliary input #7 pin 10000 Counter #0 output 10001 Counter #1 output 10010 Counter #2 output 10011 Counter #3 output 10100 Counter #4 output 10101 Counter #5 output 10110 Counter #6 output pin 10111 Counter #7 output pin 11000 IP Bus clock divide by 1 prescaler 11001 IP Bus clock divide by 2 prescaler 11010 IP Bus clock divide by 4 prescaler 11011 IP Bus clock divide by 8 prescaler 11100 IP Bus clock divide by 16 prescaler 11101 IP Bus clock divide by 32 prescaler 11110 IP Bus clock divide by 64 prescaler 11111 IP Bus clock divide by 128 prescaler
8 ONCE	Count once This bit selects continuous or one shot counting mode.

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description
	0 Count repeatedly. 1 Count until compare and then stop. When output mode 4h is used, the counter re-initializes after reaching the COMP1 value and continues to count to the COMP2 value then stops.
9 LENGTH	Count Length This bit determines whether the counter counts to the compare value and then re-initializes itself to the value specified in the LOAD, CMPLD1, or CMPLD2 registers, or the counter continues counting past the compare value, to the binary roll over. 0 Continue counting to roll over. 1 Count until compare, then reinitialize. The value that the counter is reinitialized with depends on the settings of CLC1 and CLC2. If neither of these indicates the counter is to be loaded from one of the CMPLD registers, then the LOAD register is used to reinitialize the counter upon matching either COMP register. If one of CLC1 or CLC2 indicates that the counter is to be loaded from one of the CMPLD registers, then the counter will reinitialize to the value in the appropriate CMPLD register upon a match with the appropriate COMP register. If both of the CLC1 and CLC2 fields indicate that the counter is to be loaded from the CMPLD registers, then CMPLD1 will have priority if both compares happen at the same value. When output mode 4h is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc.
10 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 0 Count up 1 Count down
11–15 SECSRC	Secondary Count Source These bits identify the source to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the SIPS bit of the CTRL2 register. 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Counter #6 input pin 00111 Counter #7 input pin 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Auxiliary input #2 pin 01011 Auxiliary input #3 pin 01100 Auxiliary input #4 pin 01101 Auxiliary input #5 pin 01110 Auxiliary input #6 pin 01111 Auxiliary input #7 pin 10000 Counter #0 output 10001 Counter #1 output

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description	
	10010	Counter #2 output
	10011	Counter #3 output
	10100	Counter #4 output
	10101	Counter #5 output
	10110	Counter #6 output pin
	10111	Counter #7 output pin
	11000-11111	Reserved

41.4.9 Channel n Control Register 2 (ETIMER_CH*n*_CTRL2)

Address: 0h base + 10h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read	OEN	RDNT	INPUT	VAL	0	COFRC	COINIT	
Write					FORCE			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	SIPS	PIPS	OPS	MSTR	OUTMODE			
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL2 field descriptions

Field	Description
0 OEN	Output Enable This bit determines the direction of the external pin. 0 The external pin is configured as an input. 1 OFLAG output signal will be driven on the external pin. Other timer channels using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.
1 RDNT	Redundant Channel Enable This bit enables redundant channel checking between adjacent channels (0 and 1, 2 and 3, 4 and 5 , 6 and 7). When this bit is clear, the RCF bit in this channel cannot be set. When this bit is set, the RCF bit will be set by a miscompare between the OFLAG of this channel and the OFLAG of its redundant adjacent channel which will cause the output of this channel to go inactive (logic 0 prior to consideration of the OPS bit). 0 Disable redundant channel checking. 1 Enable redundant channel checking.
2 INPUT	External input signal This read only bit reflects the current state of the signal selected via SECSRC after application of the SIPS bit and filtering.

Table continues on the next page...

ETIMER_CH*n*_CTRL2 field descriptions (continued)

Field	Description
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs.
4 FORCE	Force the OFLAG output This write only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit always reads as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if OUTMODE is 0000 (software controlled). Setting this bit while the OUTMODE is a different value may yield unpredictable results.
5 COFRC	Co-channel OFLAG Force This bit enables the compare from another channel within the module to force the state of this counter's OFLAG output signal. 0 Other channels cannot force the OFLAG of this channel. 1 Other channels may force the OFLAG of this channel.
6–7 COINIT	Co-channel Initialization These bits enable another channel within the module to force the re-initialization of this channel when the other channel has an active compare event. 00 Other channels cannot force re-initialization of this channel. 01 Other channels may force a re-initialization of this channel's counter using the LOAD reg. 10 Other channels may force a re-initialization of this channel's counter with the CMPLD2 reg when this channel is counting down or the CMPLD1 reg when this channel is counting up. 11 Reserved
8 SIPS	Secondary Source Input Polarity Select This bit inverts the polarity of the signal selected by the SECSRC bits. 0 True polarity. 1 Inverted polarity.
9 PIPS	Primary Source Input Polarity Select This bit inverts the polarity of the signal selected by the PRISRC bits. This only applies if the signal selected by PRISRC is not the prescaled IP Bus clock. 0 True polarity. 1 Inverted polarity.
10 OPS	Output Polarity Select. This bit inverts the OFLAG output signal polarity. 0 True polarity. 1 Inverted polarity.
11 MSTR	Master Mode This bit enables the compare function's output to be broadcasted to the other channels in the module. The compare signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs.

Table continues on the next page...

ETIMER_CH*n*_CTRL2 field descriptions (continued)

Field	Description
	0 Disable broadcast of compare events from this channel. 1 Enable broadcast of compare events from this channel.
12–15 OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 0000 Software controlled 0001 Clear OFLAG output on successful compare (COMP1 or COMP2) 0010 Set OFLAG output on successful compare (COMP1 or COMP2) 0011 Toggle OFLAG output on successful compare (COMP1 or COMP2) 0100 Toggle OFLAG output using alternating compare registers 0101 Set on compare with COMP1, cleared on secondary source input edge 0110 Set on compare with COMP2, cleared on secondary source input edge 0111 Set on compare, cleared on counter roll-over 1000 Set on successful compare on COMP1, clear on successful compare on COMP2 1001 Asserted while counter is active, cleared when counter is stopped. 1010 Asserted when counting up, cleared when counting down. 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Enable gated clock output while counter is active

41.4.10 Channel n Control Register 3 (ETIMER_CH*n*_CTRL3)

Address: 0h base + 12h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved		ROC		Reserved			
Reset	0	0	0	0	1	1	1	1

Bit	8	9	10	11	12	13	14	15
Read	C2FCNT			C1FCNT			DBGEN	
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL3 field descriptions

Field	Description
0 Reserved	This field is reserved. Always write the reset value to this field.
1–2 ROC	Reload on Capture These bits enable the capture function to cause the counter to be reloaded from the LOAD register.

Table continues on the next page...

ETIMER_CH*n*_CTRL3 field descriptions (continued)

Field	Description
	00 Do not reload the counter on a capture event. 01 Reload the counter on a capture 1 event. 10 Reload the counter on a capture 2 event. 11 Reload the counter on both a capture 1 event and a capture 2 event.
3–7 Reserved	This field is reserved. Always write the reset value to this field.
8–10 C2FCNT	Capture 2 FIFO count This field reflects the number of words in the CAPT2 FIFO.
11–13 C1FCNT	Capture 1 FIFO count This field reflects the number of words in the CAPT1 FIFO.
14–15 DBGEN	Debug Actions Enable These bits allow the counter channel to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt channel counter during debug mode. 10 Force OFLAG to logic 0 (prior to consideration of the OPS bit) during debug mode. 11 Both halt counter and force OFLAG to 0 during debug mode.

41.4.11 Channel n Status Register (ETIMER_CH*n*_STS)

Address: 0h base + 14h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						0	RCF	ICF2	ICF1	IEHF	IELF	TOF	TCF2	TCF1	TCF
Write								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_STS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RCF	Redundant Channel Flag This field is set to 1 when there is a miscompare between this channel's OFLAG value and the OFLAG value of the corresponding redundant channel. Corresponding channels are grouped together in the following pairs: 0 and 1, 2 and 3, 4 and 5 , or 6 and 7 . This field can only be set to 1 if the RDNT bit is set. This field is cleared by writing a 1 to this field. **NOTE:** This field is functional only in the even channels, i.e., 0, 2, 4 , and 6. In the odd channels, this field cannot create an interrupt.

Table continues on the next page...

ETIMER_CH*n*_STS field descriptions (continued)

Field	Description
8 ICF2	Input Capture 2 Flag This bit is set when an input capture event (as defined by CPT2MODE) occurs while the word count of the CAPT2 FIFO meets or exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF2DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF2DE is set (DMA).
9 ICF1	Input Capture 1 Flag This bit is set when an input capture event (as defined by CPT1MODE) occurs while the word count of the CAPT1 FIFO meets or exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF1DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF1DE is set (DMA).
10 IEHF	Input Edge High Flag This bit is set when a positive input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
11 IELF	Input Edge Low Flag This bit is set when a negative input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
12 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value FFFFh or 0000h (depending on count direction). This bit is cleared by writing a one to this bit location.
13 TCF2	Timer Compare 2 Flag This bit is set when a successful compare occurs with COMP2. This bit is cleared by writing a one to this bit location.
14 TCF1	Timer Compare 1 Flag This bit is set when a successful compare occurs with COMP1. This bit is cleared by writing a one to this bit location.
15 TCF	This bit is set when a successful compare occurs. This bit is cleared by writing a one to this bit location.

41.4.12 Channel n Interrupt and DMA Enable Register (ETIMER_CH*n*_INTDMA)

Address: 0h base + 16h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read Write	ICF2DE	ICF1DE	CMPLD2DE	CMPLD1DE	0		Reserved	RCFIE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ICF2IE	ICF1IE	IEHFIE	IELFIE	TOFIE	TCF2IE	TCF1IE	TCFIE
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_INTDMA field descriptions

Field	Description
0 ICF2DE	Input Capture 2 Flag DMA Enable Setting this bit enables DMA read requests for CAPT2 when the ICF2 bit is set. Clearing this bit clears any pending DMA requests. Do not set both this bit and the ICF2IE bit.
1 ICF1DE	Input Capture 1 Flag DMA Enable Setting this bit enables DMA read requests for CAPT1 when the ICF1 bit is set. Clearing this bit clears any pending DMA requests. Do not set both this bit and the ICF1IE bit.
2 CMPLD2DE	Comparator Load Register 2 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD2 register whenever data is transferred out of the CMPLD2 reg into either the CNTR, COMP1, or COMP2 registers.
3 CMPLD1DE	Comparator Load Register 1 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD1 register whenever data is transferred out of the CMPLD1 reg into either the CNTR, COMP1, or COMP2 registers.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. Always write the reset value to this field.
7 RCFIE	Redundant Channel Flag Interrupt Enable Setting this bit enables interrupts when the RCF bit is set. This bit is only in even channels (0, 2, 4 , and 6).
8 ICF2IE	Input Capture 2 Flag Interrupt Enable Setting this bit enables interrupts when the ICF2 bit is set. Do not set both this bit and the ICF2DE bit.
9 ICF1IE	Input Capture 1 Flag Interrupt Enable Setting this bit enables interrupts when the ICF1 bit is set. Do not set both this bit and the ICF1DE bit.
10 IEHFIE	Input Edge High Flag Interrupt Enable Setting this bit enables interrupts when the IEHF bit is set.
11 IELFIE	Input Edge Low Flag Interrupt Enable Setting this bit enables interrupts when the IELF bit is set.
12 TOFIE	Timer Overflow Flag Interrupt Enable Setting this bit enables interrupts when the TOF bit is set.
13 TCF2IE	Timer Compare 2 Flag Interrupt Enable Setting this bit enables interrupts when the TCF2 bit is set.
14 TCF1IE	Timer Compare 1 Flag Interrupt Enable Setting this bit enables interrupts when the TCF1 bit is set.
15 TCFIE	Timer Compare Flag Interrupt Enable Setting this bit enables interrupts when the TCF bit is set.

41.4.13 Channel n Comparator Load Register 1 (ETIMER_CH*n*_CMPLD1)

This read/write register is the preload value for the COMP1 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in Usage of Compare Load Registers .

Address: 0h base + 18h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CMPLD1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CMPLD1 field descriptions

Field	Description
0–15 CMPLD1	Preload value for the COMP1 register

41.4.14 Channel n Comparator Load Register 2 (ETIMER_CH*n*_CMPLD2)

This read/write register is the preload value for the COMP2 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in Usage of Compare Load Registers

Address: 0h base + 1Ah offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CMPLD2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CMPLD2 field descriptions

Field	Description
0–15 CMPLD2	Preload value for the COMP2 register

41.4.15 Channel n Compare and Capture Control Register (ETIMER_CH*n*_CCCTRL)

Address: 0h base + 1Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read Write	CLC2				CLC1			CMPMODE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CPT2MODE		CPT1MODE		CFWM		ONESHOT	ARM
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CCCTRL field descriptions

Field	Description
0–2 CLC2	Compare Load Control 2 These bits control when COMP2 is preloaded. It also controls the loading of CNTR. 000 Never preload 001 Reserved 010 Load COMP2 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP2 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP2 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP2 with CMPLD2 upon successful compare with the value in COMP2. 110 Load CNTR with CMPLD2 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD2 upon successful compare with the value in COMP2.
3–5 CLC1	Compare Load Control 1 These bits control when COMP1 is preloaded. It also controls the loading of CNTR. 000 Never preload 001 Reserved 010 Load COMP1 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP1 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP1 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP1 with CMPLD2 upon successful compare with the value in COMP2. 110 Load CNTR with CMPLD1 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD1 upon successful compare with the value in COMP2.
6–7 CMPMODE	Compare Mode These bits control when the COMP1 and COMP2 registers are used in regards to the counting direction. 00 COMP1 register is used when the counter is counting up; COMP2 register is used when the counter is counting up. 01 COMP1 register is used when the counter is counting down; COMP2 register is used when the counter is counting up.

Table continues on the next page...

ETIMER_CH*n*_CCCTRL field descriptions (continued)

Field	Description
	10 COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting down. 11 COMP1 register is used when the counter is counting down; COMP2 register is used when the counter is counting down.
8–9 CPT2MODE	Capture 2 Mode Control These bits control the operation of the CAPT2 register as well as the operation of the ICF2 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges. 11 Capture any edge.
10–11 CPT1MODE	Capture 1 Mode Control These bits control the operation of the CAPT1 register as well as the operation of the ICF1 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges. 11 Capture any edge.
12–13 CFWM	Capture FIFO Water Mark This field represents the water mark level for the CAPT1 and CAPT2 FIFOs. The capture flags, ICF1 and ICF2, won't be set until the word count of the corresponding FIFO is greater than this water mark level.
14 ONESHOT	One Shot Capture Mode This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and capture circuit 1 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and the ARM bit is cleared. No further captures will be performed until the ARM bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARM bit is then cleared.
15 ARM	Arm Capture Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). 0 Input capture operation is disabled. 1 Input capture operation as specified by the CPT1MODE and CPT2MODE bits is enabled.

41.4.16 Channel n Input Filter Register (ETIMER_CH*n*_FILT)

The FILT register programs the values for the filtering of the corresponding input without regard for the fact that any eTimer channel can use the input as a count source. The FILT register is also used to control the filtering of the corresponding auxiliary input if one exists.

The FILT_PER value should be set such that the sampling period is larger the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of: (((FILT_CNT + 3) x FILT_PER) + 2) peripheral clock periods.

Address: 0h base + 1Eh offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FILT_CNT			FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_FILT field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in this section.
8–15 FILT_PER	Input Filter Sample Period These bits represent the sampling period (in peripheral clock cycles) of the eTimer input signal. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is $00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in this section.

41.4.17 Channel Enable Register (ETIMER_ENBL)

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								ENBL							
Write																
Reset	0	0	0	0	0	0	0	0	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- ENBL field: Reset value depends on field width. Unimplemented bits (channels) will reset to 0. See the chip-specific information for the eTimer module.

ETIMER_ENBL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate channels. If an ENBL bit is set, then the corresponding channel will start counting as soon as the CNTMODE field has a value other than 000. When an ENBL bit is clear, the corresponding channel maintains its current value. **NOTE:** The width of this field is equal to the number of channels a particular eTimer instance supports. If the chip has more than one eTimer instance, refer to the chip-specific information for the eTimer module for the configuration of each eTimer instance. 0 Timer channel is disabled 1 Timer channel is enabled. This is the reset value.

41.4.18 DMA Request 0 Select Register (ETIMER_DREQ0)

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DREQ0_EN				0				0				DREQ0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_DREQ0 field descriptions

Field	Description
0 DREQ0_EN	DMA Request Enable

Table continues on the next page...

ETIMER_DREQ0 field descriptions (continued)

Field	Description
	Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ0	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request 00110: Channel 1 CMPLD1 DMA write request 00111: Channel 1 CMPLD2 DMA write request 01000: Channel 2 CAPT1 DMA read request 01001: Channel 2 CAPT2 DMA read request 01010: Channel 2 CMPLD1 DMA write request 01011: Channel 2 CMPLD2 DMA write request 01100: Channel 3 CAPT1 DMA read request 01101: Channel 3 CAPT2 DMA read request 01110: Channel 3 CMPLD1 DMA write request 01111: Channel 3 CMPLD2 DMA write request 10000: Channel 4 CAPT1 DMA read request 10001: Channel 4 CAPT2 DMA read request 10010: Channel 4 CMPLD1 DMA write request 10011: Channel 4 CMPLD2 DMA write request 10100: Channel 5 CAPT1 DMA read request 10101: Channel 5 CAPT2 DMA read request 10110: Channel 5 CMPLD1 DMA write request 10111: Channel 5 CMPLD2 DMA write request 11000: Channel 6 CAPT1 DMA read request 11001: Channel 6 CAPT2 DMA read request 11010: Channel 6 CMPLD1 DMA write request

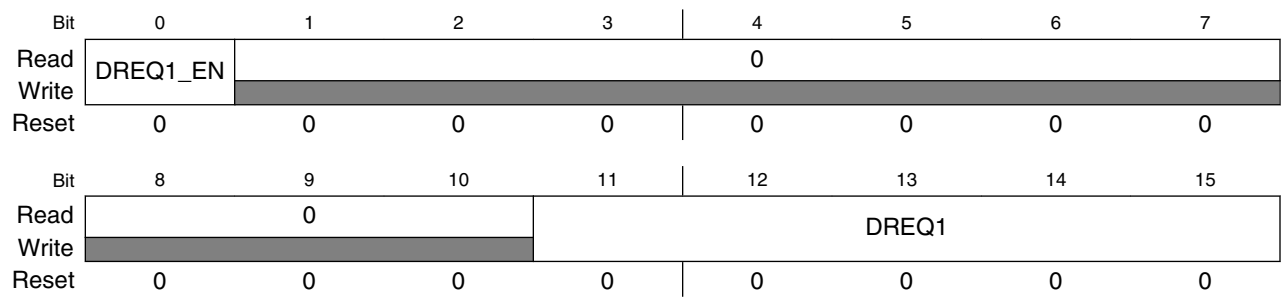
Table continues on the next page...

ETIMER_DREQ0 field descriptions (continued)

Field	Description
	11011: Channel 6 CMPLD2 DMA write request
	11100: Channel 7 CAPT1 DMA read request
	11101: Channel 7 CAPT2 DMA read request
	11110: Channel 7 CMPLD1 DMA write request
	11111: Channel 7 CMPLD2 DMA write request

41.4.19 DMA Request 1 Select Register (ETIMER_DREQ1)

Address: 0h base + 112h offset = 112h



ETIMER_DREQ1 field descriptions

Field	Description
0 DREQ1_EN	DMA Request Enable Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ1	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request

Table continues on the next page...

ETIMER_DREQ1 field descriptions (continued)

Field	Description
	00110: Channel 1 CMPLD1 DMA write request
	00111: Channel 1 CMPLD2 DMA write request
	01000: Channel 2 CAPT1 DMA read request
	01001: Channel 2 CAPT2 DMA read request
	01010: Channel 2 CMPLD1 DMA write request
	01011: Channel 2 CMPLD2 DMA write request
	01100: Channel 3 CAPT1 DMA read request
	01101: Channel 3 CAPT2 DMA read request
	01110: Channel 3 CMPLD1 DMA write request
	01111: Channel 3 CMPLD2 DMA write request
	10000: Channel 4 CAPT1 DMA read request
	10001: Channel 4 CAPT2 DMA read request
	10010: Channel 4 CMPLD1 DMA write request
	10011: Channel 4 CMPLD2 DMA write request
	10100: Channel 5 CAPT1 DMA read request
	10101: Channel 5 CAPT2 DMA read request
	10110: Channel 5 CMPLD1 DMA write request
	10111: Channel 5 CMPLD2 DMA write request
	11000: Channel 6 CAPT1 DMA read request
	11001: Channel 6 CAPT2 DMA read request
	11010: Channel 6 CMPLD1 DMA write request
	11011: Channel 6 CMPLD2 DMA write request
	11100: Channel 7 CAPT1 DMA read request
	11101: Channel 7 CAPT2 DMA read request
	11110: Channel 7 CMPLD1 DMA write request
	11111: Channel 7 CMPLD2 DMA write request

41.5 Functional Description

This section provides a functional description of eTimer.

41.5.1 General

Each channel has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
    - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the channels within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs is selectable.

The primary output of each channel is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is programmable.

Any channel can be assigned as a "Master". A master's compare signal can be broadcasted to the other channels within the module. The other channels can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a Master channel's compare event occurs.

41.5.2 Counting Modes

The selected external signals are sampled at the eTimer's base clock rate and then run through a transition detector. The maximum count rate is one-half of the eTimer's base clock rate when using an external signal. Internal clock sources can be used to clock the counters at the eTimer's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CNTMODE field in the CTRL1 register is cleared when the count terminates.

41.5.2.1 STOP Mode

If the CNTMODE field is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

41.5.2.2 COUNT Mode

If the CNTMODE field is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting the PIPS bit, then the negative edge of the selected external input signal is counted.

See CASCADE-COUNT Mode through VARIABLE-FREQUENCY PWM Mode for additional capabilities of this operating mode.

41.5.2.3 EDGE-COUNT Mode

If the CNTMODE field is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

41.5.2.4 GATED-COUNT Mode

If the CNTMODE field is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the SIPS bit, then the counter will count while the selected secondary input is low.

41.5.2.5 QUADRATURE-COUNT Mode

If the CNTMODE field is set to 100, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signals provides both count and direction information.

The following figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

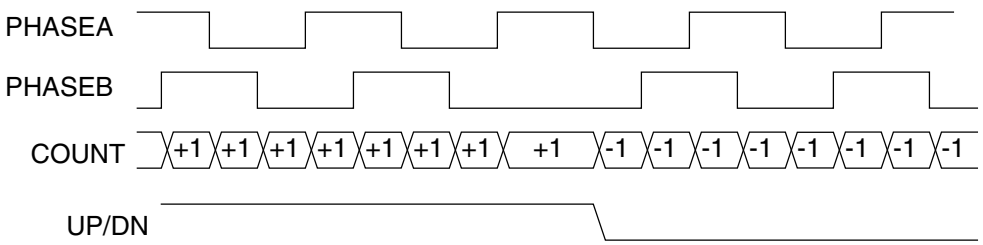


Figure 41-3. Quadrature Incremental Position Encoder

41.5.2.6 SIGNED-COUNT Mode

If the CNTMODE field is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

41.5.2.7 TRIGGERED-COUNT Mode

If the CNTMODE field is set to '110', the counter will begin counting the primary clock source after a positive transition (negative if SIPS = 1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. Subsequent secondary positive input transitions will continue to restart and stop the counting until a compare event occurs.

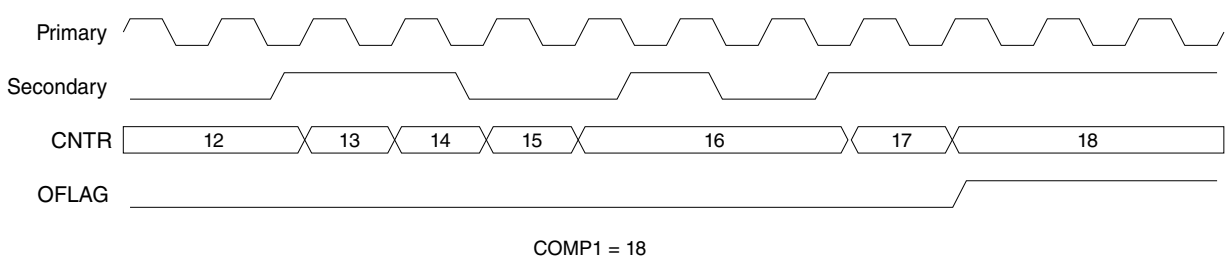


Figure 41-4. Triggered Count Mode (Length=1)

41.5.2.8 ONE-SHOT Mode

If the CNTMODE field is set to '110', and the counter is set to reinitialize at a compare event (LENGTH =1), and the OFLAG OUTMODE is set to '0101' (cleared on init, set on compare), the counter works in a "One-Shot Mode". An external events causes the counter to count, when terminal count is reached, the output is asserted. This "delayed" output can be used to provide timing delays.

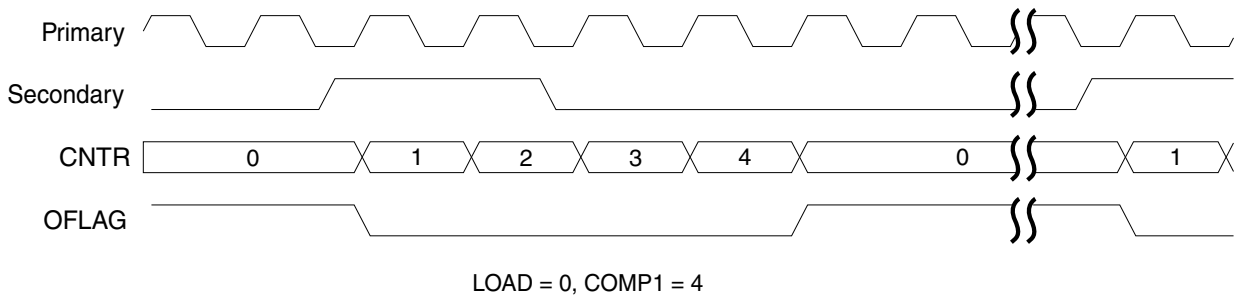


Figure 41-5. One-Shot Mode (Length=1)

41.5.2.9 CASCADE-COUNT Mode

If the CNTMODE field is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This "Cascade" or "Daisy-Chained" mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to two counters may be cascaded to create a 32 bit wide synchronous counter.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective HOLD registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the HOLD registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a "ripple" mode, where higher order counters will transition a clock later than a purely synchronous design.

Note

A channel can be cascaded with any other channel, but you can't cascade more than 2 channels together. You can create separate cascades of pairs of channels. For example, you can cascade channels 0 and 1 and separately cascade channels 6 and 5. You can't cascade channels 0, 1, and 5.

41.5.2.10 PULSE-OUTPUT Mode

If the counter is setup for CNTMODE = 001, and the OFLAG OUTMODE is set to '1111' (gated clock output), and the ONCE bit is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note

This does not work if the PRISRC is set to 11000 (IP_bus/1).

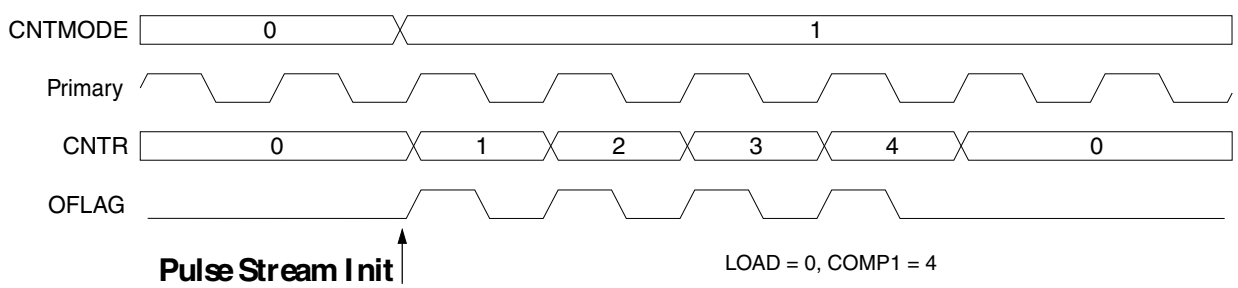


Figure 41-6. Pulse Output Mode

41.5.2.11 FIXED-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count through roll-over (LENGTH = 0), continuous count (ONCE = 0) and the OFLAG OUTMODE is '0111' (set on compare, cleared on counter roll-over) then the counter output yields a Pulse Width Modulated

(PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

41.5.2.12 VARIABLE-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count till compare (LENGTH = 1), continuous count (ONCE = 0) and the OFLAG OUTMMODE is '0100' (toggle OFLAG and alternate compare registers) then the counter output yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

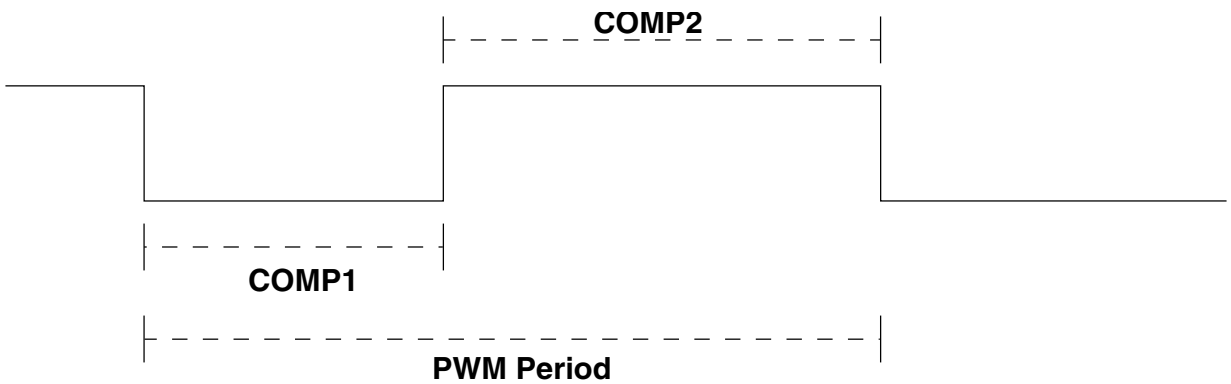
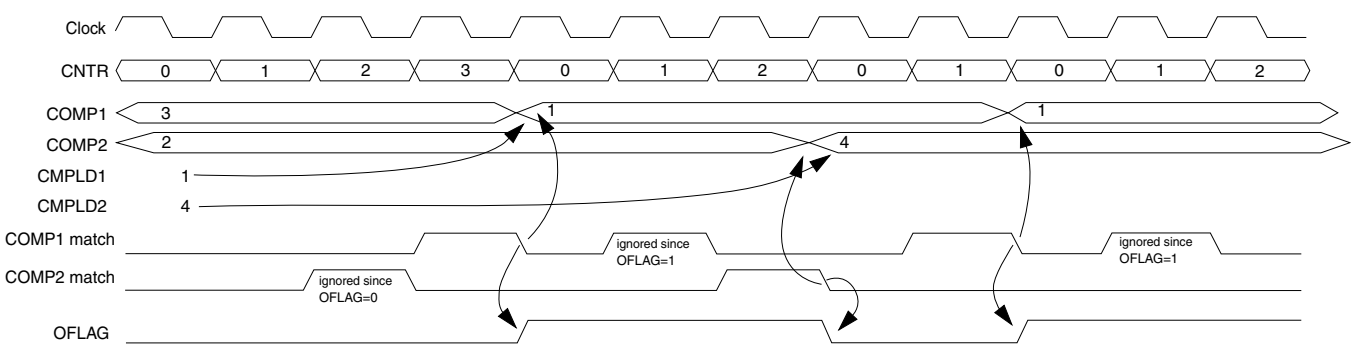


Figure 41-7. Variable PWM Waveform



CLC1=010, load COMP1 when CNTR=COMP1
CLC2=101, load COMP2 when CNTR=COMP2

Figure 41-8. Variable Frequency PWM Mode Timing

41.5.2.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If the output mode is set to 0100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. COMP1 is used when OFLAG=0 and COMP2 is used when OFLAG=1. OFLAG can be forced to a value using CTRL2[FORCE] and CTRL2[VAL].

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: CNTR = COMPx, *not* CNTR > COMP1 or CNTR < COMP2.

The use of the CMPLD1 and CMPLD2 registers to preload compare values will help to minimize this problem.

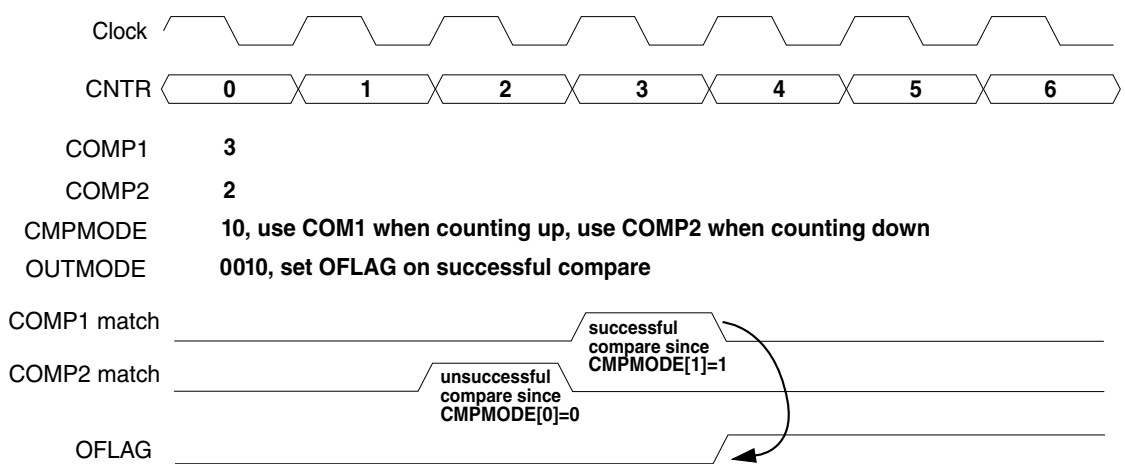


Figure 41-9. Compare Register and OFLAG Timing

41.5.2.14 Usage of Compare Load Registers

The CMPLD1, CMPLD2 and CCCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. A compare register can be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there is the possibility that the counter may have already counted past the new compare value by the time the compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are updated in hardware in the same way the counter register is re-initialized to the value stored in the LOAD register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See Figure 41-8.

41.5.2.15 MODULO COUNTING Mode

To create a modulo counter using COMP1 and COMP2 as the counter boundaries (instead of $0000 and $FFFF), set the registers in the following manner. Set CNTMODE to either 100 (quadrature count mode) or 101 (count with direction mode). Use count through roll-over (LENGTH = 0) and continuous count (ONCE = 0). Set COMP1 and CMPLD1 to the upper boundary value. Set COMP2 and CMPLD2 to the lower boundary value. Set CMPMODE = 10 (COMP1 is used when counting up and COMP2 is used when counting down). Set CLC2 = 110 (load CNTR with value of CMPLD2 on COMP1 compare) and CLC1 = 111 (load CNTR with value of CMPLD1 on COMP2 compare).

41.5.2.16 Compare Register and OFLAG Operation

The compare flags and output flag are registered signals and can only change state after the counter reached the compare value. This means that, if the counter is continuously counting, then the compare flag is set on the count after the counter reaches the compare value. When counting is not continuous (such as when using a prescaler or when counting edges on the primary input), the compare flag or OFLAG are not updated until the counter is about to transition to the next count. This means that if the compare value is 4, then the flags will be changed on the same clock edge that transitions the counter from 4 to its next value.

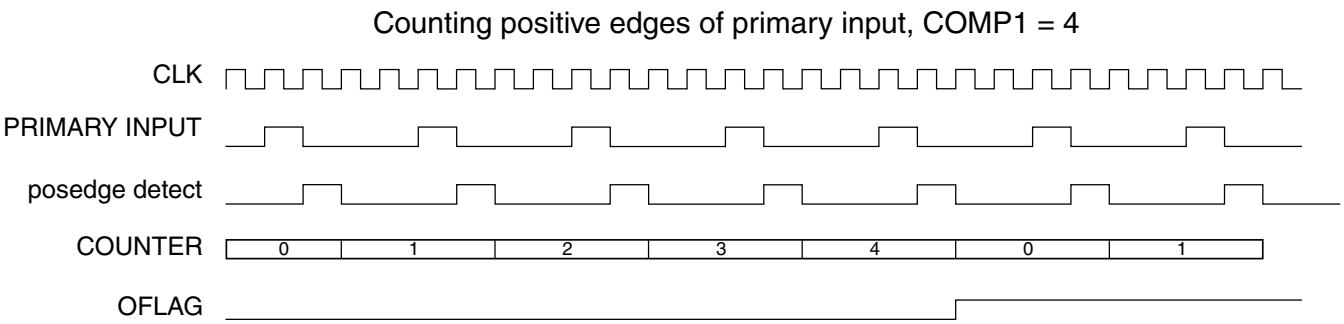


Figure 41-10. OFLAG timing during non-continuous counting

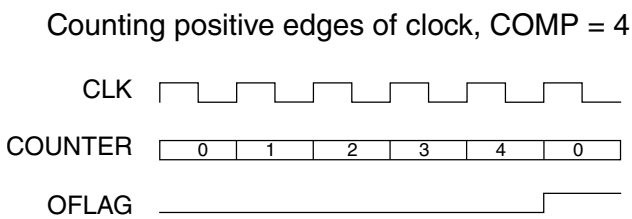


Figure 41-11. OFLAG timing during continuous counting

41.5.3 Other Features

This section describes other features of eTimer.

41.5.3.1 Redundant OFLAG Checking

This mode allows the user to bundle two timer functions generating any pattern to compare their resulting OFLAG behaviors (output signal).

The redundant mode is used to support online checks for reliability and functional safety reasons. Whenever a mismatch between the two adjacent channels occurs, it is reported via an interrupt to the core and the two outputs are put into their inactive states. An error is flagged via the RCF flag.

This feature can be tested by forcing a transition on one of the OFLAGs using the VAL and FORCE bits of the channel.

41.5.3.2 Loopback Checking

This mode is always available in that one channel can generate an OFLAG while another channel uses the first channels OFLAG as its input to be measured and verified to be as expected.

41.5.3.3 Input Capture Mode

Input capture is used to measure pulse width (by capturing the counter value on two successive input edges) or waveform period (by capturing the counter value on two consecutive rising edges or two consecutive falling edges). The Capture Registers store a copy of the counter's value when an input edge (positive, negative, or both) is detected. The type of edge to be captured by each circuit is determined by the CPT1MODE and CPT2MODE bits whose functionality is listed in CPT1MODE. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

41.5.3.4 Master/Slave Mode

Any timer channel can be assigned as a Master (MSTR = 1). A Master's compare signal can be broadcasted to the other channels within the module. The other counters can be configured to reinitialize their counters (COINIT = 1) and/or force their OFLAG output signals (COFRC = 1) to predetermined values when a Master counter compare event occurs.

41.6 Resets

The eTimer module can only be reset by a full system reset. This forces all registers to their reset state and clears the OFLAG signals if they are asserted. The counters will be turned off until the settings in the CTRL registers are changed.

41.7 Clocks

Each timer channel receives its own copy of the peripheral bus clock. This allows for better power management by turning off the clock to unused channels. Each bit of the ENBL register uses the clock for that corresponding channel. The DREQn registers use the clock for channel, 0 but this clock can be turned off after these registers have been programmed.

41.8 Interrupts

The following can generate interrupts within the eTimer.

- Each channel can generate an interrupt from several sources

The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 41-4. Interrupt summary

Core interrupt	Interrupt flag	Interrupt enable	Name	Description
Channels 0-n	TCF	TCFIE	Compare interrupt	Compare of counter and related compare register
	TCF1	TCF1IE	Compare 1 interrupt	Compare of the counter and COMP1 register
	TCF2	TCF2IE	Compare 2 interrupt	Compare of the counter and COMP2 register
	TOF	TOFIE	Overflow interrupt	Generated on counter roll-over or roll-under
	IELF	IELFIE	Input Low Edge interrupt	Falling edge of the secondary input signal
	IEHF	IEHFIE	Input High Edge interrupt	Rising edge of the secondary input signal
	ICF1	ICF1IE	Input Capture 1 interrupt	Input capture event for CAPT1
	ICF2	ICF2IE	Input Capture 2 interrupt	Input capture event for CAPT2
Redundant Channel Checking	RCF	RCFIE	Redundant Channel Fault interrupt	Miscompare with redundant channel

41.9 DMA

Each channel can request a DMA read access for each of the capture registers and a DMA write request for each of the compare preload registers. The DREQ registers select amongst these 32 DMA request sources to generate the 4 top level DMA request outputs.

Table 41-5. DMA Summary

DMA Request	DMA Enable	Name	Description
Channels 0-n	ICF1DE	CAPT1 read request	CAPT1 contains a value
	ICF2DE	CAPT2 read request	CAPT2 contains a value
	CMPLD1DE	CMPLD1 write request	CMPLD1 needs an update
	CMPLD2DE	CMPLD2 write request	CMPLD2 needs an update

Chapter 42
Motor Control Pulse Width Modulator (FlexPWM)

42.1 Chip-specific FlexPWM information

The following table show the inter-module connection of the FlexPWM for the Motor Control related peripherals for the device.

Table 42-1. Motor Control peripheral interconnects

Input Module	Input Signal	Driving Module	Driver Signal
FLEXPWM	EXT_FORCE	eTIMER_2	T1
FLEXPWM	CLOCK	eTIMER_1	T1
FLEXPWM	EXTA	-	-
FLEXPWM	EXTB	-	-
FLEXPWM	EXT_SWITCH	-	-

42.1.1 FlexPWM chip-specific register reset values

FlexPWM_FSTS reset values for this chip is 0x0F0F when SIUL2_MSCR and SIUL2_IMCR registers are not programmed.

42.2 Introduction

This section contains features, modes of operation, and block diagrams of the FlexPWM.

42.2.1 Overview

The pulse width modulator module (FlexPWM) contains one or more PWM submodules, each capable of controlling a single half-bridge power stage. One or more fault channels may also be included.

This PWM is capable of controlling most motor types: AC induction motors (ACIM), Permanent Magnet AC motors (PMAC), both brushless (BLDC) and brush DC motors (BDC), switched (SRM) and variable reluctance motors (VRM), and stepper motors.

42.2.2 Features

The FlexPWM module provides the following features:

- 16 bits of resolution for center, edge aligned, and asymmetrical PWMs
- PWM outputs capable of operating as complementary pairs or independent channels
- Can accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Synchronization to external hardware or other PWM supported
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a "Force Out" event
- PWMX pin can optionally output a third PWM signal from each submodule
- PWMX channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions

- Capture is supported only for X channels not A and B channels
- Dual edge capture functionality
- Option to supply the source for each complementary PWM signal pair from SOC external or internal signals

42.2.3 Modes of operation

Care must be exercised when using this module in certain chip operating modes. Some motors (such 3-phase AC motors) require regular software updates for proper operation. Failure to do so could result in destroying the motor or inverter. Because of this, PWM outputs are placed in their inactive states in STOP mode, and optionally under debug mode. PWM outputs are reactivated (assuming they were active to begin with) when these modes are exited.

Table 42-2. Modes when PWM operation is restricted

Mode	Description
STOP	Peripheral and CPU clocks are stopped. PWM outputs are driven inactive.
DEBUG	CPU and peripheral clocks continue to run, but CPU maybe stalled for periods of time. PWM outputs are driven inactive as a function of the DBGEN bit.

42.2.4 Block Diagrams

This section contains block diagrams of the FlexPWM.

42.2.4.1 Module level

Note that not all signals are shown in this diagram.

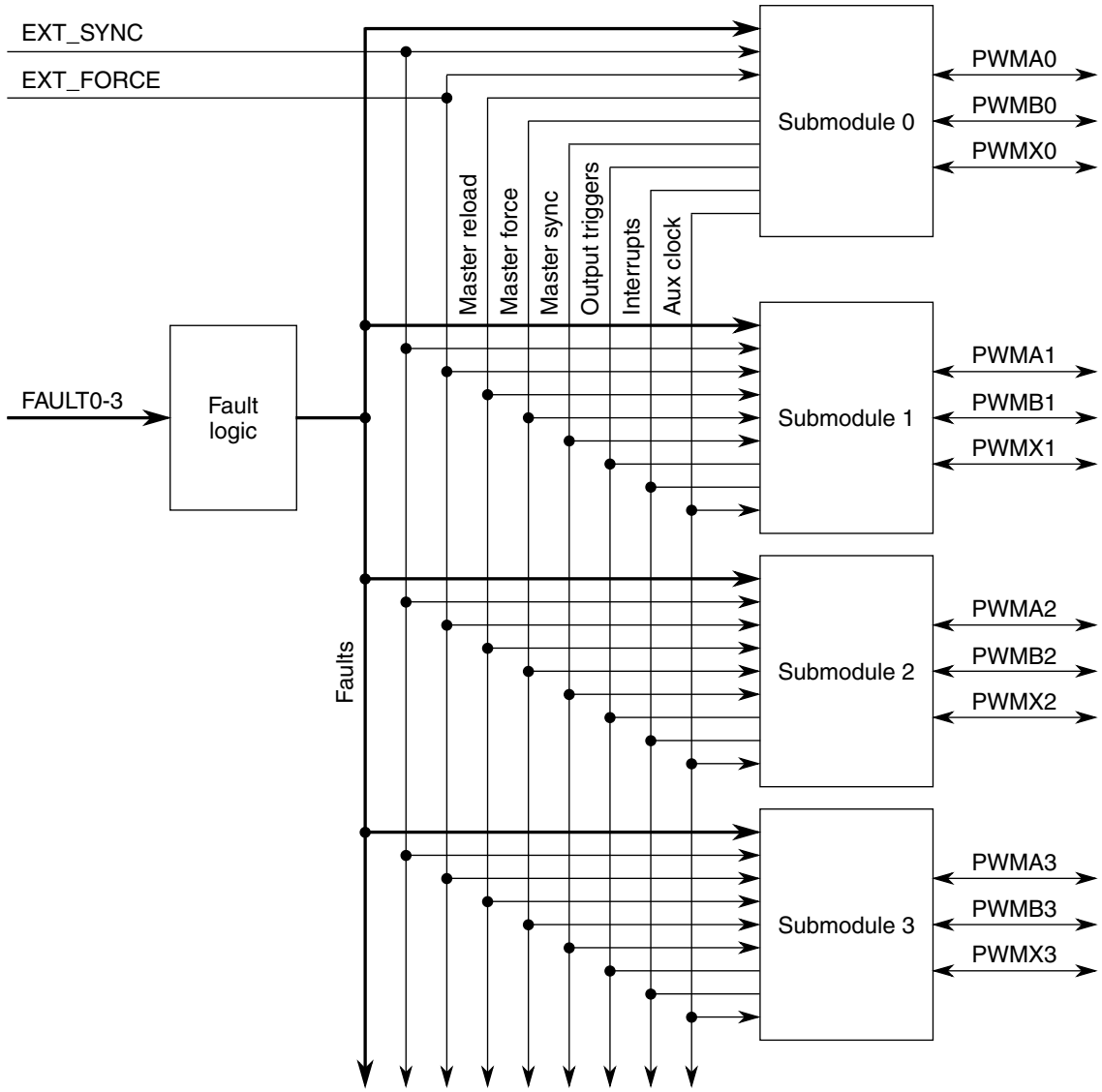


Figure 42-1. PWM block diagram

42.2.4.2 PWM submodule

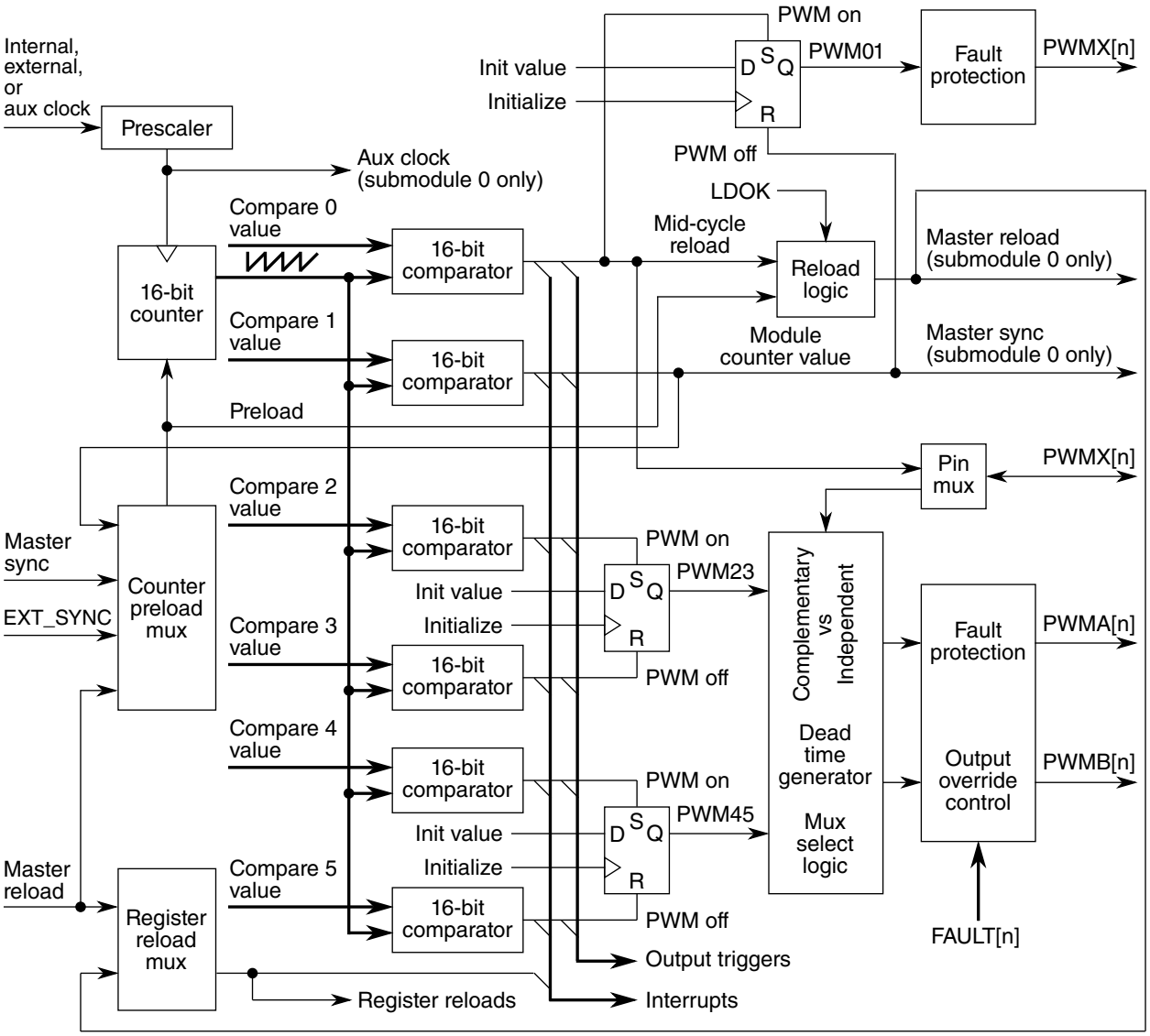


Figure 42-2. PWM submodule block diagram

42.3 External Signal Descriptions

The PWM module has external pins named PWMA[n], PWMB[n], PWMX[n], FAULT[n], EXT_SYNC, EXT_FORCE, EXTA[n], and EXTB[n]. The PWM module also has an external clock input called EXT_CLK and an output triggers bus.

42.3.1 PWMA[n] and PWMB[n] - External PWM Pair

These pins are the output pins of the PWM submodules. They can be independent PWM signals or a complementary pair.

42.3.2 PWMX[n] - Auxiliary PWM signal

These pins are the auxiliary output pins of the PWM submodules. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or they can be used to generate the IPOL bit during deadtime correction.

42.3.3 FAULT[n] - Fault inputs

These are input pins for disabling selected PWM outputs.

42.3.4 EXT_SYNC - External Synchronization Signal

This input signal allows a source external to the PWM to initialize the PWM counter. In this manner the PWM can be synchronized to external circuitry.

42.3.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner the PWM can be synchronized to external circuitry. An example would be to simultaneously switch all of the PWM outputs on a commutation boundary for trapezoidal control of a BLDC motor. The boundary can be established via external logic or an on-chip timer.

42.3.6 EXTA[n] and EXTB[n] - Alternate PWM Control Signals

These pins allow an alternate source to control the PWMA and PWMB outputs although typically, the EXTA input will be used for the generation of a complementary pair. Typical connections include ADC results registers, TMR outputs, GPIO inputs, and comparator outputs.

42.3.7 OUT_TRIG_EVEN[n] and OUT_TRIG_ODD[n] - Output Triggers

These outputs allow the PWM submodules to control timing of the ADC conversions. See Submodule n Output Trigger Control Register (FlexPWM_SUB*n*_TCTRL) for a description of how to enable these outputs and how the compare registers match up to the output triggers.

42.3.8 EXT_CLK - External Clock Enable

This on-chip input signal allows an on-chip source external to the PWM (typically a Timer) to control the PWM clocking. In this manner the PWM can be synchronized to the Timer. This signal must be generated synchronously to the PWM's clock since it is not resynchronized in the PWM.

42.3.9 EXT_SWITCH - External Switch Signal

This input signal allows a source external to the PWM to control the PWM generation source within submodule1. This signal only functions on submodule1 and selects the source of the generated PWM signal to be either submodule1 or submodule0.

42.4 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level. There are a set of registers for each PWM submodule, for the configuration logic, and for each Fault channel.

Registers are only accessible using 16- or 32-bit accesses. Byte accesses are not allowed.

Submodule registers are repeated for each PWM submodule. The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is 50h. This is the base address of the PWM module plus an offset based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same offset.

The base address of the configuration registers is equal to the base address of the FlexPWM plus as offset of 140h.

The base address of the fault logic is equal to the base address of the FlexPWM plus an offset of 14Ch.

NOTE

A write access to the read-only Capture Value and Capture Value Cycle registers will result in a bus error.

NOTE

The space at offset 14Ah is reserved. Accessing this location does not result in a transfer error. Read accesses return 0h, and write accesses have no effect.

FlexPWM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Submodule n Counter Register (FlexPWM_SUB0_CNT)	16	R	0000h	42.4.1/1574
2	Submodule n Initial Count Register (FlexPWM_SUB0_INIT)	16	R/W	0000h	42.4.2/1575
4	Submodule n Control 2 Register (FlexPWM_SUB0_CTRL2)	16	R/W	0000h	42.4.3/1575
6	Submodule n Control 1 Register (FlexPWM_SUB0_CTRL1)	16	R/W	0400h	42.4.4/1578
8	Submodule n Value Register 0 (FlexPWM_SUB0_VAL0)	16	R/W	0000h	42.4.5/1579
A	Submodule n Value Register 1 (FlexPWM_SUB0_VAL1)	16	R/W	0000h	42.4.6/1580
C	Submodule n Value Register 2 (FlexPWM_SUB0_VAL2)	16	R/W	0000h	42.4.7/1581
E	Submodule n Value Register 3 (FlexPWM_SUB0_VAL3)	16	R/W	0000h	42.4.8/1582
10	Submodule n Value Register 4 (FlexPWM_SUB0_VAL4)	16	R/W	0000h	42.4.9/1582
12	Submodule n Value Register 5 (FlexPWM_SUB0_VAL5)	16	R/W	0000h	42.4.10/ 1583
18	Submodule n Output Control Register (FlexPWM_SUB0_OCTRL)	16	R/W	See section	42.4.11/ 1584
1A	Submodule n Status Register (FlexPWM_SUB0_STS)	16	R/W	0000h	42.4.12/ 1585
1C	Submodule n Interrupt Enable Register (FlexPWM_SUB0_INTEN)	16	R/W	0000h	42.4.13/ 1587
1E	Submodule n DMA Enable Register (FlexPWM_SUB0_DMAEN)	16	R/W	0000h	42.4.14/ 1588
20	Submodule n Output Trigger Control Register (FlexPWM_SUB0_TCTRL)	16	R/W	0000h	42.4.15/ 1589
22	Submodule n Fault Disable Mapping Register (FlexPWM_SUB0_DISMAP)	16	R/W	FFFFh	42.4.16/ 1590
24	Submodule n Deadtime Count Register 0 (FlexPWM_SUB0_DTCNT0)	16	R/W	07FFh	42.4.17/ 1592
26	Submodule n Deadtime Count Register 1 (FlexPWM_SUB0_DTCNT1)	16	R/W	07FFh	42.4.18/ 1592

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30	Submodule n Capture Control X Register (FlexPWM_SUB0_CAPTCTRLX)	16	R/W	0000h	42.4.19/ 1593
32	Submodule n Capture Compare X Register (FlexPWM_SUB0_CAPTCMPX)	16	R/W	0000h	42.4.20/ 1595
34	Submodule n Capture Value 0 Register (FlexPWM_SUB0_CVAL0)	16	R	0000h	42.4.21/ 1595
36	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB0_CVAL0CYC)	16	R	0000h	42.4.22/ 1596
38	Submodule n Capture Value 1 Register (FlexPWM_SUB0_CVAL1)	16	R	0000h	42.4.23/ 1596
3A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB0_CVAL1CYC)	16	R	0000h	42.4.24/ 1597
50	Submodule n Counter Register (FlexPWM_SUB1_CNT)	16	R	0000h	42.4.1/1574
52	Submodule n Initial Count Register (FlexPWM_SUB1_INIT)	16	R/W	0000h	42.4.2/1575
54	Submodule n Control 2 Register (FlexPWM_SUB1_CTRL2)	16	R/W	0000h	42.4.3/1575
56	Submodule n Control 1 Register (FlexPWM_SUB1_CTRL1)	16	R/W	0400h	42.4.4/1578
58	Submodule n Value Register 0 (FlexPWM_SUB1_VAL0)	16	R/W	0000h	42.4.5/1579
5A	Submodule n Value Register 1 (FlexPWM_SUB1_VAL1)	16	R/W	0000h	42.4.6/1580
5C	Submodule n Value Register 2 (FlexPWM_SUB1_VAL2)	16	R/W	0000h	42.4.7/1581
5E	Submodule n Value Register 3 (FlexPWM_SUB1_VAL3)	16	R/W	0000h	42.4.8/1582
60	Submodule n Value Register 4 (FlexPWM_SUB1_VAL4)	16	R/W	0000h	42.4.9/1582
62	Submodule n Value Register 5 (FlexPWM_SUB1_VAL5)	16	R/W	0000h	42.4.10/ 1583
68	Submodule n Output Control Register (FlexPWM_SUB1_OCTRL)	16	R/W	See section	42.4.11/ 1584
6A	Submodule n Status Register (FlexPWM_SUB1_STS)	16	R/W	0000h	42.4.12/ 1585
6C	Submodule n Interrupt Enable Register (FlexPWM_SUB1_INTEN)	16	R/W	0000h	42.4.13/ 1587
6E	Submodule n DMA Enable Register (FlexPWM_SUB1_DMAEN)	16	R/W	0000h	42.4.14/ 1588
70	Submodule n Output Trigger Control Register (FlexPWM_SUB1_TCTRL)	16	R/W	0000h	42.4.15/ 1589
72	Submodule n Fault Disable Mapping Register (FlexPWM_SUB1_DISMAP)	16	R/W	FFFFh	42.4.16/ 1590
74	Submodule n Deadtime Count Register 0 (FlexPWM_SUB1_DTCNT0)	16	R/W	07FFh	42.4.17/ 1592
76	Submodule n Deadtime Count Register 1 (FlexPWM_SUB1_DTCNT1)	16	R/W	07FFh	42.4.18/ 1592
80	Submodule n Capture Control X Register (FlexPWM_SUB1_CAPTCTRLX)	16	R/W	0000h	42.4.19/ 1593
82	Submodule n Capture Compare X Register (FlexPWM_SUB1_CAPTCMPX)	16	R/W	0000h	42.4.20/ 1595

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
84	Submodule n Capture Value 0 Register (FlexPWM_SUB1_CVAL0)	16	R	0000h	42.4.21/ 1595
86	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB1_CVAL0CYC)	16	R	0000h	42.4.22/ 1596
88	Submodule n Capture Value 1 Register (FlexPWM_SUB1_CVAL1)	16	R	0000h	42.4.23/ 1596
8A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB1_CVAL1CYC)	16	R	0000h	42.4.24/ 1597
A0	Submodule n Counter Register (FlexPWM_SUB2_CNT)	16	R	0000h	42.4.1/1574
A2	Submodule n Initial Count Register (FlexPWM_SUB2_INIT)	16	R/W	0000h	42.4.2/1575
A4	Submodule n Control 2 Register (FlexPWM_SUB2_CTRL2)	16	R/W	0000h	42.4.3/1575
A6	Submodule n Control 1 Register (FlexPWM_SUB2_CTRL1)	16	R/W	0400h	42.4.4/1578
A8	Submodule n Value Register 0 (FlexPWM_SUB2_VAL0)	16	R/W	0000h	42.4.5/1579
AA	Submodule n Value Register 1 (FlexPWM_SUB2_VAL1)	16	R/W	0000h	42.4.6/1580
AC	Submodule n Value Register 2 (FlexPWM_SUB2_VAL2)	16	R/W	0000h	42.4.7/1581
AE	Submodule n Value Register 3 (FlexPWM_SUB2_VAL3)	16	R/W	0000h	42.4.8/1582
B0	Submodule n Value Register 4 (FlexPWM_SUB2_VAL4)	16	R/W	0000h	42.4.9/1582
B2	Submodule n Value Register 5 (FlexPWM_SUB2_VAL5)	16	R/W	0000h	42.4.10/ 1583
B8	Submodule n Output Control Register (FlexPWM_SUB2_OCTRL)	16	R/W	See section	42.4.11/ 1584
BA	Submodule n Status Register (FlexPWM_SUB2_STS)	16	R/W	0000h	42.4.12/ 1585
BC	Submodule n Interrupt Enable Register (FlexPWM_SUB2_INTEN)	16	R/W	0000h	42.4.13/ 1587
BE	Submodule n DMA Enable Register (FlexPWM_SUB2_DMAEN)	16	R/W	0000h	42.4.14/ 1588
C0	Submodule n Output Trigger Control Register (FlexPWM_SUB2_TCTRL)	16	R/W	0000h	42.4.15/ 1589
C2	Submodule n Fault Disable Mapping Register (FlexPWM_SUB2_DISMAP)	16	R/W	FFFFh	42.4.16/ 1590
C4	Submodule n Deadtime Count Register 0 (FlexPWM_SUB2_DTCNT0)	16	R/W	07FFh	42.4.17/ 1592
C6	Submodule n Deadtime Count Register 1 (FlexPWM_SUB2_DTCNT1)	16	R/W	07FFh	42.4.18/ 1592
D0	Submodule n Capture Control X Register (FlexPWM_SUB2_CAPTCTRLX)	16	R/W	0000h	42.4.19/ 1593
D2	Submodule n Capture Compare X Register (FlexPWM_SUB2_CAPTCMPX)	16	R/W	0000h	42.4.20/ 1595
D4	Submodule n Capture Value 0 Register (FlexPWM_SUB2_CVAL0)	16	R	0000h	42.4.21/ 1595
D6	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB2_CVAL0CYC)	16	R	0000h	42.4.22/ 1596

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D8	Submodule n Capture Value 1 Register (FlexPWM_SUB2_CVAL1)	16	R	0000h	42.4.23/ 1596
DA	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB2_CVAL1CYC)	16	R	0000h	42.4.24/ 1597
F0	Submodule n Counter Register (FlexPWM_SUB3_CNT)	16	R	0000h	42.4.1/1574
F2	Submodule n Initial Count Register (FlexPWM_SUB3_INIT)	16	R/W	0000h	42.4.2/1575
F4	Submodule n Control 2 Register (FlexPWM_SUB3_CTRL2)	16	R/W	0000h	42.4.3/1575
F6	Submodule n Control 1 Register (FlexPWM_SUB3_CTRL1)	16	R/W	0400h	42.4.4/1578
F8	Submodule n Value Register 0 (FlexPWM_SUB3_VAL0)	16	R/W	0000h	42.4.5/1579
FA	Submodule n Value Register 1 (FlexPWM_SUB3_VAL1)	16	R/W	0000h	42.4.6/1580
FC	Submodule n Value Register 2 (FlexPWM_SUB3_VAL2)	16	R/W	0000h	42.4.7/1581
FE	Submodule n Value Register 3 (FlexPWM_SUB3_VAL3)	16	R/W	0000h	42.4.8/1582
100	Submodule n Value Register 4 (FlexPWM_SUB3_VAL4)	16	R/W	0000h	42.4.9/1582
102	Submodule n Value Register 5 (FlexPWM_SUB3_VAL5)	16	R/W	0000h	42.4.10/ 1583
108	Submodule n Output Control Register (FlexPWM_SUB3_OCTRL)	16	R/W	See section	42.4.11/ 1584
10A	Submodule n Status Register (FlexPWM_SUB3_STS)	16	R/W	0000h	42.4.12/ 1585
10C	Submodule n Interrupt Enable Register (FlexPWM_SUB3_INTEN)	16	R/W	0000h	42.4.13/ 1587
10E	Submodule n DMA Enable Register (FlexPWM_SUB3_DMAEN)	16	R/W	0000h	42.4.14/ 1588
110	Submodule n Output Trigger Control Register (FlexPWM_SUB3_TCTRL)	16	R/W	0000h	42.4.15/ 1589
112	Submodule n Fault Disable Mapping Register (FlexPWM_SUB3_DISMAP)	16	R/W	FFFFh	42.4.16/ 1590
114	Submodule n Deadtime Count Register 0 (FlexPWM_SUB3_DTCNT0)	16	R/W	07FFh	42.4.17/ 1592
116	Submodule n Deadtime Count Register 1 (FlexPWM_SUB3_DTCNT1)	16	R/W	07FFh	42.4.18/ 1592
120	Submodule n Capture Control X Register (FlexPWM_SUB3_CAPTCTRLX)	16	R/W	0000h	42.4.19/ 1593
122	Submodule n Capture Compare X Register (FlexPWM_SUB3_CAPTCMPX)	16	R/W	0000h	42.4.20/ 1595
124	Submodule n Capture Value 0 Register (FlexPWM_SUB3_CVAL0)	16	R	0000h	42.4.21/ 1595
126	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB3_CVAL0CYC)	16	R	0000h	42.4.22/ 1596
128	Submodule n Capture Value 1 Register (FlexPWM_SUB3_CVAL1)	16	R	0000h	42.4.23/ 1596
12A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB3_CVAL1CYC)	16	R	0000h	42.4.24/ 1597

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
140	Output Enable Register (FlexPWM_OUTEN)	16	R/W	0000h	42.4.25/ 1597
142	Mask Register (FlexPWM_MASK)	16	R/W	0000h	42.4.26/ 1599
144	Software Controlled Output Register (FlexPWM_SWCOUT)	16	R/W	0000h	42.4.27/ 1601
146	Deadtime Source Select Register (FlexPWM_DTSRCSEL)	16	R/W	0000h	42.4.28/ 1602
148	Master Control Register (FlexPWM_MCTRL)	16	R/W	0000h	42.4.29/ 1604
14C	Fault Control Register (FlexPWM_FCTRL)	16	R/W	0000h	42.4.30/ 1606
14E	Fault Status Register (FlexPWM_FSTS)	16	R/W	See section	42.4.31/ 1608
150	Fault Filter Register (FlexPWM_FFILT)	16	R/W	0000h	42.4.32/ 1610
152	Fault Control 2 Register (FlexPWM_FCTRL2)	16	R/W	0000h	42.4.33/ 1610

42.4.1 Submodule n Counter Register (FlexPWM_SUB*n*_CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 0h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CNT field descriptions

Field	Description
0–15 CNT	Count State of the signed 16-bit submodule counter

42.4.2 Submodule n Initial Count Register (FlexPWM_SUB*n*_INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of INIT_SEL) or when FORCE is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. This register cannot be written when LDOK is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 2h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_INIT field descriptions

Field	Description
0–15 INIT	Initial count Initial count value for the PWM in clock periods

42.4.3 Submodule n Control 2 Register (FlexPWM_SUB*n*_CTRL2)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 4h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read / Write	DBGEN	Reserved	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
Write		FORCE						
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CTRL2 field descriptions

Field	Description
0 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
1 Reserved	This field is reserved. Always write the reset value to this field.
2 INDEP	Independent or Complementary Pair Operation This bit determines if the PWMA and PWMB channels will be independent PWMs or a complementary PWM pair. 0 PWMA and PWMB form a complementary PWM pair. 1 PWMA and PWMB outputs are independent PWMs.
3 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
4 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
5 PWMX_INIT	PWMX Initial Value This read/write bit determines the initial value for PWMX and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
6–7 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWMX) causes initialization. 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0.

Table continues on the next page...

FlexPWM_SUB*n*_CTRL2 field descriptions (continued)

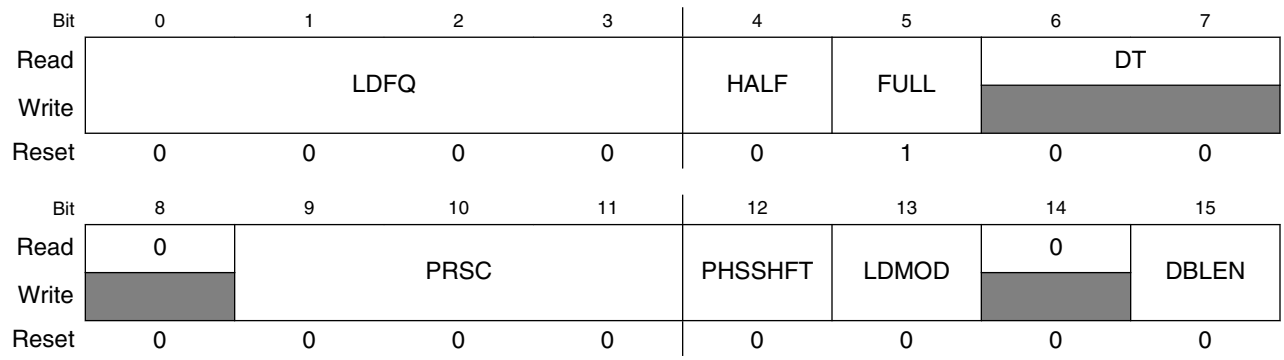
Field	Description
	10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 EXT_SYNC causes initialization.
8 FRCEN	Force Initialization Enable This bit allows the FORCE_OUT signal to initialize the counter without regard to the signal selected by INIT_SEL. This is a software controlled initialization. 0 Initialization from a Force Out event is disabled. 1 Initialization from a Force Out event is enabled.
9 FORCE	Force Initialization If the FORCE_SEL bits are set to 000, writing a 1 to this bit results in a Force Out event. This causes the following actions to be taken: • The PWMA and PWMB output pins will assume values based on the SEL23 and SEL45 bits. • If the FRCEN bit is set, the counter value will be initialized with the INIT register value.
10–12 FORCE_SEL	Force Source Select This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, FORCE, from this submodule is used to force updates. 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011 The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 The local sync signal from this submodule is used to force updates. 101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 reserved
13 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, the LDOK bit in submodule 0 should be used since the local LDOK bit will be ignored. 0 The local RELOAD signal is used to reload registers. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
14–15 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The peripheral clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 reserved

42.4.4 Submodule n Control 1 Register (FlexPWM_SUB*n*_CTRL1)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 6h offset + (80d × i), where i=0d to 3d



FlexPWM_SUB*n*_CTRL1 field descriptions

Field	Description
0–3 LDFQ	Load Frequency These buffered read/write bits select the PWM load frequency. The PWM reload frequency is at every LDFQ+1 PWM opportunities. Reset clears the LDFQ bits, selecting loading every PWM opportunity. A PWM opportunity is determined by the HALF and FULL bits. **NOTE:** The LDFQx bits take effect when the current load cycle is complete regardless of the state of the LDOK bit. Reading the LDFQx bits reads the buffered values and not necessarily the values currently in effect.
4 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the SUB*n*_VAL0 register and does not have to be half way through the PWM cycle. 0 Half-cycle reloads disabled. 1 Half-cycle reloads enabled.
5 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the SUB*n*_VAL1 register. Either the HALF or FULL bit must be set in order to move the buffered data into the registers used by the PWM generators. If both the HALF and FULL bits are set, then reloads can occur twice per cycle. 0 Full-cycle reloads disabled. 1 Full-cycle reloads enabled.
6–7 DT	Deadtime These read only bits reflect the sampled values of the PWMX input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.

Table continues on the next page...

FlexPWM_SUB*n*_CTRL1 field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CLK_SEL. **NOTE:** Reading the PRSCx bits reads the buffered values and not necessarily the values currently in effect. The PRSCx bits take effect at the beginning of the next PWM cycle and only when the load okay bit, LDOK, is set or LDMOD is set. This field cannot be written when LDOK is set. 000 PWM clock frequency = f_{CLK} 001 PWM clock frequency = f_{CLK} / 2 010 PWM clock frequency = f_{CLK} / 4 011 PWM clock frequency = f_{CLK} / 8 100 PWM clock frequency = f_{CLK} / 16 101 PWM clock frequency = f_{CLK} / 32 110 PWM clock frequency = f_{CLK} / 64 111 PWM clock frequency = f_{CLK} / 128
12 PHSSHFT	Phase Shift Mode Enable This read/write bit is only functional in submodule 0. It is used to enable the EXT_SWITCH input to select the source of the generated PWM23/PWM45 signals within submodule 0. If this enable is clear or the EXT_SWITCH signal is 0, then the PWM23/PWM45 signals generated within submodule 0 are used by the force out logic in submodule 0. If this enable is set and the EXT_SWITCH signal is also set, then the PWM23/PWM45 signals from submodule 1 are used by the force out logic in submodule 0. 0 Phase shifted mode is disabled. The EXT_SWITCH input is ignored. Normal operation. 1 Phase shifted mode is enabled. The EXT_SWITCH input is used to select generated PWM source for force out logic in submodule 0.
13 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if LDOK is set. 1 Buffered registers of this submodule are loaded and take effect immediately upon LDOK being set.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior. 0 Double switching disabled. 1 Double switching enabled.

42.4.5 Submodule n Value Register 0 (FlexPWM_SUB*n*_VAL0)

The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in clock periods. This value also defines when the PWMX signal is set and the local sync signal is reset. This register is not byte accessible.

NOTE

The SUB*n*_VAL0 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL0 cannot be written when LDOK is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 8h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL0 field descriptions

Field	Description
0–15 VAL0	Value 0 Mid-cycle reload point for the PWM in clock periods

42.4.6 Submodule n Value Register 1 (FlexPWM_SUB*n*_VAL1)

The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter will reload itself with the contents of the INIT register and assert the local sync signal while resetting PWMX. This register is not byte accessible.

NOTE

The SUB*n*_VAL1 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL1 cannot be written when LDOK is set. Reading VAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

NOTE

If SUB*n*_VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. The PWMX output will be low for a minimum of one count every cycle after the count reaches VAL1. When the Master Sync signal (only

originated by the Local Sync from sub-module 0) is used to control the timer period, the SUB*n*_VAL1 register can be free for other functions such as PWM generation without duty cycle limitation.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL1 field descriptions

Field	Description
0–15 VAL1	Value 1 Modulo count value (maximum count) for the submodule counter

42.4.7 Submodule n Value Register 2 (FlexPWM_SUB*n*_VAL2)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL2 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL2 cannot be written when LDOK is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Ch offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL2 field descriptions

Field	Description
0–15 VAL2	Value 2 Count value to set PWM23 high

42.4.8 Submodule n Value Register 3 (FlexPWM_SUB*n*_VAL3)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL3 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL3 cannot be written when LDOK is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Eh offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL3 field descriptions

Field	Description
0–15 VAL3	Value 3 Count value to set PWM23 low

42.4.9 Submodule n Value Register 4 (FlexPWM_SUB*n*_VAL4)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL4 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL4 cannot be written when

LDOK is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 10h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL4 field descriptions

Field	Description
0–15 VAL4	Count value to set PWM45 high

42.4.10 Submodule n Value Register 5 (FlexPWM_SUB*n*_VAL5)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL5 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL5 cannot be written when LDOK is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 12h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL5 field descriptions

Field	Description
0–15 VAL5	Count value to set PWM45 low

42.4.11 Submodule n Output Control Register (FlexPWM_SUB*n*_OCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 18h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0	0	PWMX_IN	0		POLA	POLB	POLX
Write								
Reset	0	0	*	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		PWMAFS		PWMBFS		PWMXFS	
Write								
Reset	0	0	0	0	0	0	0	0

* Notes:
- PWMX_IN field: Undefined

FlexPWM_SUB*n*_OCTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 PWMX_IN	PWMX Input This read only bit shows the logic value currently being driven into the PWMX input.
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 POLA	PWMA Output Polarity This bit inverts the PWMA output polarity. 0 PWMA output not inverted. A high level on the PWMA pin represents the "on" or "active" state. 1 PWMA output inverted. A low level on the PWMA pin represents the "on" or "active" state.
6 POLB	PWMB Output Polarity This bit inverts the PWMB output polarity. 0 PWMB output not inverted. A high level on the PWMB pin represents the "on" or "active" state. 1 PWMB output inverted. A low level on the PWMB pin represents the "on" or "active" state.
7 POLX	PWMX Output Polarity This bit inverts the PWMX output polarity.

Table continues on the next page...

FlexPWM_SUB*n*_OCTRL field descriptions (continued)

Field	Description
	0 PWMX output not inverted. A high level on the PWMX pin represents the "on" or "active" state. 1 PWMX output inverted. A low level on the PWMX pin represents the "on" or "active" state.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–11 PWMAFS	PWMA Fault State These bits determine the fault state for the PWMA output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
12–13 PWMBFS	PWMB Fault State These bits determine the fault state for the PWMB output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
14–15 PWMXFS	PWMX Fault State These bits determine the fault state for the PWMX output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

42.4.12 Submodule n Status Register (FlexPWM_SUB*n*_STS)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	RUF	REF	RF	0				CFX1	CFX0	CMPF					
Write			w1c	w1c					w1c	w1c	w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_STS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 RUF	Registers Updated Flag This read-only flag is set when SUB*n*_INIT, SUB*n*_VAL*x*, or SUB*n*_CTRL1[PRSC] has been written resulting in non-coherent data in the set of double-buffered registers. Clear RUF by a proper reload sequence consisting of a reload signal while LDOK = 1. Reset clears RUF. 0 No register update has occurred since last reload. 1 At least one of the double buffered registers has been updated since the last reload.
2 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while LDOK is 0 and the double buffered registers are in a non-coherent state (RUF = 1). Clear REF by writing a logic one to the REF bit. Reset clears REF. 0 No reload error occurred. 1 Reload signal occurred with non-coherent data and LDOK = 0.
3 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear RF by writing a logic one to the RF bit when VALDE is clear (non-DMA mode). RF can also be cleared by the DMA done signal when VALDE is set (DMA mode). Reset clears RF. 0 No new reload cycle since last RF clearing 1 New reload cycle since last RF clearing
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 CFX1	Capture Flag X1 This bit is set when the word count of the Capture X1 FIFO (CX1CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX1DE is clear (non-DMA mode) or by the DMA done signal if CX1DE is set (DMA mode). Reset clears this bit.
9 CFX0	Capture Flag X0 This bit is set when the word count of the Capture X0 FIFO (CX0CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX0DE is clear (non-DMA mode) or by the DMA done signal if CX0DE is set (DMA mode). Reset clears this bit.
10–15 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the SUB*n*_VAL*x* registers. Clear these bits by writing a 1 to a bit position.

CMPF bit	Value register
0	SUB*n*_VAL0
1	SUB*n*_VAL1
2	SUB*n*_VAL2
3	SUB*n*_VAL3
4	SUB*n*_VAL4
5	SUB*n*_VAL5

Table continues on the next page...

FlexPWM_SUB*n*_STS field descriptions (continued)

Field	Description
	0 No compare event has occurred for a particular VAL*x* value. 1 A compare event has occurred for a particular VAL*x* value.

42.4.13 Submodule n Interrupt Enable Register (FlexPWM_SUB*n*_INTEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Ch offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0		REIE	RIE	Reserved			
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CX1IE	CX0IE	CMPIE					
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_INTEN field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag (REF) to generate CPU interrupt requests. Reset clears RIE. 0 REF CPU interrupt requests disabled 1 REF CPU interrupt requests enabled
3 RIE	Reload Interrupt Enable This read/write bit enables the reload flag (RF) to generate CPU interrupt requests. Reset clears RIE. 0 RF CPU interrupt requests disabled 1 RF CPU interrupt requests enabled
4–7 Reserved	This field is reserved. Always write the reset value to this field.
8 CX1IE	Capture X 1 Interrupt Enable This bit allows the CFX1 flag to create an interrupt request to the CPU. Do not set both this bit and the CX1DE bit. 0 Interrupt request disabled for CFX1. 1 Interrupt request enabled for CFX1.

Table continues on the next page...

FlexPWM_SUB*n*_INTEN field descriptions (continued)

Field	Description
9 CX0IE	Capture X 0 Interrupt Enable This bit allows the CFX0 flag to create an interrupt request to the CPU. Do not set both this bit and the CX0DE bit. 0 Interrupt request disabled for CFX0. 1 Interrupt request enabled for CFX0.
10–15 CMPIE	Compare Interrupt Enables These bits enable the CMPF flags to cause a compare interrupt request to the CPU. **CMPIE bit** / **Value register** 0 / SUB*n*_VAL0 1 / SUB*n*_VAL1 2 / SUB*n*_VAL2 3 / SUB*n*_VAL3 4 / SUB*n*_VAL4 5 / VAL5 0 The corresponding CMPF bit will not cause an interrupt request. 1 The corresponding CMPF bit will cause an interrupt request.

42.4.14 Submodule n DMA Enable Register (FlexPWM_SUB*n*_DMAEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Eh offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0						VALDE	FAND
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CAPTDE		Reserved				CX1DE	CX0DE
Write								
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_DMAEN field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 VALDE	Value Registers DMA Enable This read/write bit enables DMA write requests for the SUB*n*_VAL*x* registers when RF is set. Reset clears VALDE. Clearing this field clears any pending DMA requests. 0 DMA write requests disabled 1 DMA write requests for the SUB*n*_VAL*x* registers enabled
7 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the CAPTDE field when it is set to watermark mode (CAPTDE = 01). While the CAxDE, CBxDE, and CXxDE bits determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 0 Selected FIFO watermarks are OR'ed together. 1 Selected FIFO watermarks are AND'ed together.
8–9 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00 Read DMA requests disabled. 01 Exceeding a FIFO watermark sets the DMA read request. This requires at least 1 of the CA1DE, CA0DE, CB1DE, CB0DE, CX1DE, or CX0DE bits to also be set in order to determine which watermark(s) the DMA request is sensitive to. 10 A local sync (VAL1 matches counter) sets the read DMA request. 11 A local reload (RF being set) sets the read DMA request.
10–13 Reserved	This field is reserved. Always write the reset value to this field.
14 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when CFX1 is set. Reset clears CX1DE. Clearing this field clears any pending DMA requests. Do not set both this bit and the CX1IE bit.
15 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when CFX0 is set. Reset clears CX0DE. Clearing this field clears any pending DMA requests. Do not set both this bit and the CX0IE bit.

42.4.15 Submodule n Output Trigger Control Register (FlexPWM_SUB*n*_TCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 20h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										OUT_TRIG_EN					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_TCTRL field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of OUT_TRIG_EVEN and OUT_TRIG_ODD outputs based on the counter value matching the value in one or more of the SUB*n*_VAL*x* registers. VAL0, VAL2, and VAL4 are used to generate OUT_TRIG_EVEN and VAL1, VAL3, and VAL5 are used to generate OUT_TRIG_ODD. The OUT_TRIGx signals are only asserted as long as the counter value matches the VAL*x* value, therefore up to six triggers can be generated (three each on OUT_TRIG_EVEN and OUT_TRIG_ODD) per PWM cycle per submodule.

OUT_TRIG_EN bit	Value register
0	SUB*n*_VAL0
1	SUB*n*_VAL1
2	SUB*n*_VAL2
3	SUB*n*_VAL3
4	SUB*n*_VAL4
5	SUB*n*_VAL5

0 OUT_TRIGx will not set when the counter value matches the VAL*x* value.
1 OUT_TRIGx will set when the counter value matches the VAL*x* value.

42.4.16 Submodule n Fault Disable Mapping Register (FlexPWM_SUB*n*_DISMAP)

This register determines which PWM pins are disabled by the fault protection inputs, illustrated in the table in Fault Protection. Reset sets all of the bits in the fault disable mapping register.

NOTE

The FAULT pin still detects a signal even if the associated pin is not configured for the FlexPWM FAULT function. Read the FAULT pin value at FSTS[FFPIN].

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 22h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	1				DISX				DISB				DISA			
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DISMAP field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
4–7 DISX	PWMX Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULT*x* inputs. The PWMX output will be turned off if there is a logic 1 on a FAULT*x* input and a 1 in the corresponding bit of the DISX field. A reset sets all DISX bits.
8–11 DISB	PWMB Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULT*x* inputs. The PWMB output will be turned off if there is a logic 1 on a FAULT*x* input and a 1 in the corresponding bit of the DISB field. A reset sets all DISB bits.
12–15 DISA	PWMA Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULT*x* inputs. The PWMA output will be turned off if there is a logic 1 on a FAULT*x* input and a 1 in the corresponding bit of the DISA field. A reset sets all DISA bits.

DISX bit	Fault input
0	0
1	1
2	2
3	3

DISB bit	Fault input
0	0
1	1
2	2
3	3

DISA bit	Fault input
0	0
1	1
2	2
3	3

42.4.17 Submodule n Deadtime Count Register 0 (FlexPWM_SUB*n*_DTCNT0)

Deadtime operation is only applicable to complementary channel operation. The 11-bit value written to this register is in terms of peripheral clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count register to a default value of 0x07FF, selecting a deadtime of 2047 clock cycles. This register is not byte accessible.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 24h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					DTCNT0										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DTCNT0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 DTCNT0	Controls the deadtime during 0 to 1 transitions of the PWMA output (assuming normal polarity) in terms of clock cycles regardless of the setting of PRSC and/or CLK_SEL

42.4.18 Submodule n Deadtime Count Register 1 (FlexPWM_SUB*n*_DTCNT1)

Deadtime operation is only applicable to complementary channel operation. The 11-bit value written to this register is in terms of peripheral clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count register to a default value of 0x07FF, selecting a deadtime of 2047 clock cycles. This register is not byte accessible.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 26h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					DTCNT1										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DTCNT1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 DTCNT1	Controls the deadtime during 0 to 1 transitions of the complementary PWMB output, in terms of clock cycles regardless of the setting of PRSC and/or CLK_SEL

42.4.19 Submodule n Capture Control X Register (FlexPWM_SUB*n*_CAPTCTRLX)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 30h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	CX1CNT				CX0CNT		CFXWM	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	EDGCNTX_EN	INPSELX	EDGX1		EDGX0		ONESHOTX	ARMX
Write								
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CAPTCTRLX field descriptions

Field	Description
0–2 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO.
3–5 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO.
6–7 CFXWM	Capture X FIFOs Watermark This field represents the watermark level for capture X FIFOs. The capture flags, CFX1 and CFX0, won't be set until the word count of the corresponding FIFO is greater than this watermark level.
8 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWMX input signal. 0 Edge counter disabled and held in reset. 1 Edge counter enabled.
9 INPSELX	Input Select X

Table continues on the next page...

FlexPWM_SUB*n*_CAPTCTRLX field descriptions (continued)

Field	Description
	This bit selects between the raw PWMX input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. **NOTE:** When INPSELX = 1, the internal edge counter is enabled and the rising and/or falling edges specified by the EDGX0 and EDGX1 fields are ignored. The software must still place a value other than 00 in either or both of the EDGX0 and/or EDGX1 fields in order to enable one or both of the capture registers. 0 Raw PWMX input signal selected as source. 1 Output of edge counter/compare selected as source.
10–11 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled. 01 Capture falling edges. 10 Capture rising edges. 11 Capture any edge.
12–13 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled. 01 Capture falling edges. 10 Capture rising edges. 11 Capture any edge.
14 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
15 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). 0 Input capture operation is disabled. 1 Input capture operation as specified by the EDGX bits is enabled.

42.4.20 Submodule n Capture Compare X Register (FlexPWM_SUB*n*_CAPTCMPX)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 32h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	EDGCNTX								EDGCMPX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CAPTCMPX field descriptions

Field	Description
0–7 EDGCNTX	Edge Counter X This read only field contains the edge counter value for the PWMX input capture circuitry.
8–15 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWMX input capture circuitry.

42.4.21 Submodule n Capture Value 0 Register (FlexPWM_SUB*n*_CVAL0)

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX0 bits. This is actually a 4-deep FIFO and not a single register. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 34h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPTVAL0															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL0 field descriptions

Field	Description
0–15 CAPTVAL0	Value captured from the submodule counter

42.4.22 Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB*n*_CVAL0CYC)

This read only register stores the cycle number corresponding to the value captured in CVAL0. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. This is actually a 4 deep FIFO and not a single register.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 36h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0													CVAL0CYC		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL0CYC field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 CVAL0CYC	CVAL0 Cycle Cycle number corresponding to the value captured in CVAL0

42.4.23 Submodule n Capture Value 1 Register (FlexPWM_SUB*n*_CVAL1)

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX1 bits. This is actually a 4 deep FIFO and not a single register. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 38h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPTVAL1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL1 field descriptions

Field	Description
0–15 CAPTVAL1	Value captured from the submodule counter

42.4.24 Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB*n*_CVAL1CYC)

This read only register stores the cycle number corresponding to the value captured in CVAL1. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. This is actually a 4 deep FIFO and not a single register.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 3Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0													CVAL1CYC		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL1CYC field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 CVAL1CYC	CVAL1 Cycle Cycle number corresponding to the value captured in CVAL1

42.4.25 Output Enable Register (FlexPWM_OUTEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 140h offset = 140h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0				PWMA_EN				PWMB_EN				PWMX_EN			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_OUTEN field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 PWMA_EN	PWMA Output Enables These bits enable the PWMA outputs of each submodule. **PWMA_EN bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3 0 PWMA output disabled. 1 PWMA output enabled.
8–11 PWMB_EN	PWMB Output Enables These bits enable the PWMB outputs of each submodule. **PWMB_EN bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3 0 PWMB output disabled. 1 PWMB output enabled.
12–15 PWMX_EN	PWMX Output Enables These bits enable the PWMX outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMX pin is being used for input capture or deadtime correction. **PWMX_EN bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3

Table continues on the next page...

FlexPWM_OUTEN field descriptions (continued)

Field	Description
	0 PWMX output disabled. 1 PWMX output enabled.

42.4.26 Mask Register (FlexPWM_MASK)

NOTE

The MASKx bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading the MASK bits reads the buffered value and not necessarily the value currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 142h offset = 142h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0				MASKA				MASKB				MASKX			
Write	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_MASK field descriptions

Field	Description
0–3 UPDATE_MASK	Update Mask These self-clearing bits force the MASK* bits to be immediately updated within each of the submodules without waiting for a FORCE_OUT event. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. 0 Normal operation. MASK* bits within the submodules aren't updated until a FORCE_OUT event occurs within the submodule. 1 Immediate operation. MASK* bits within the corresponding submodules are updated on the following clock edge after setting this bit.
4–7 MASKA	PWMA Masks These bits mask the PWMA outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.

Table continues on the next page...

FlexPWM_MASK field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>MASKA bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMA output normal.
1 PWMA output masked.
</td></tr>
<tr><td>8–11
MASKB</td><td>PWMB Masks
These bits mask the PWMB outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.
<table>
<tr><th>MASKB bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMB output normal.
1 PWMB output masked.
</td></tr>
<tr><td>12–15
MASKX</td><td>PWMX Masks
These bits mask the PWMX outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.
<table>
<tr><th>MASKX bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMX output normal.
1 PWMX output masked.
</td></tr>
</table>

42.4.27 Software Controlled Output Register (FlexPWM_SWCOUT)

NOTE

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	OUT23_3	OUT45_3	OUT23_2	OUT45_2	OUT23_1	OUT45_1	OUT23_0	OUT45_0
Reset	0	0	0	0	0	0	0	0

FlexPWM_SWCOUT field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 OUT23_3	Software Controlled Output 23_3 This bit is only used when SEL23 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
9 OUT45_3	Software Controlled Output 45_3 This bit is only used when SEL45 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
10 OUT23_2	Software Controlled Output 23_2 This bit is only used when SEL23 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.

Table continues on the next page...

FlexPWM_SWCOUT field descriptions (continued)

Field	Description
11 OUT45_2	Software Controlled Output 45_2 This bit is only used when SEL45 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.
12 OUT23_1	Software Controlled Output 23_1 This bit is only used when SEL23 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
13 OUT45_1	Software Controlled Output 45_1 This bit is only used when SEL45 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
14 OUT23_0	Software Controlled Output 23_0 This bit is only used when SEL23 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
15 OUT45_0	Software Controlled Output 45_0 This bit is only used when SEL45 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

42.4.28 Deadtime Source Select Register (FlexPWM_DTSRCSEL)

NOTE

The deadtime source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 146h offset = 146h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SEL23_3		SEL45_3		SEL23_2		SEL45_2		SEL23_1		SEL45_1		SEL23_0		SEL45_0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_DTSRCSEL field descriptions

Field	Description
0–1 SEL23_3	PWM23_3 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_3 signal is used by the deadtime logic. 01 Inverted generated PWM23_3 signal is used by the deadtime logic. 10 OUT23_3 bit is used by the deadtime logic. 11 EXTA[3] signal is used by the deadtime logic.
2–3 SEL45_3	PWM45_3 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_3 signal is used by the deadtime logic. 01 Inverted generated PWM45_3 signal is used by the deadtime logic. 10 OUT45_3 bit is used by the deadtime logic. 11 EXTB[3] signal is used by the deadtime logic.
4–5 SEL23_2	PWM23_2 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_2 signal is used by the deadtime logic. 01 Inverted generated PWM23_2 signal is used by the deadtime logic. 10 OUT23_2 bit is used by the deadtime logic. 11 EXTA[2] signal is used by the deadtime logic.
6–7 SEL45_2	PWM45_2 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_2 signal is used by the deadtime logic. 01 Inverted generated PWM45_2 signal is used by the deadtime logic. 10 OUT45_2 bit is used by the deadtime logic. 11 EXTB[2] signal is used by the deadtime logic.
8–9 SEL23_1	PWM23_1 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_1 signal is used by the deadtime logic. 01 Inverted generated PWM23_1 signal is used by the deadtime logic. 10 OUT23_1 bit is used by the deadtime logic. 11 EXTA[1] signal is used by the deadtime logic.
10–11 SEL45_1	PWM45_1 Control Select

Table continues on the next page...

FlexPWM_DTSRCSEL field descriptions (continued)

Field	Description
	This field selects possible over-rides to the generated PWM45 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_1 signal is used by the deadtime logic. 01 Inverted generated PWM45_1 signal is used by the deadtime logic. 10 OUT45_1 bit is used by the deadtime logic. 11 EXTB[1] signal is used by the deadtime logic.
12–13 SEL23_0	PWM23_0 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_0 signal is used by the deadtime logic. 01 Inverted generated PWM23_0 signal is used by the deadtime logic. 10 OUT23_0 bit is used by the deadtime logic. 11 EXTA[0] signal is used by the deadtime logic.
14–15 SEL45_0	PWM45_0 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_0 signal is used by the deadtime logic. 01 Inverted generated PWM45_0 signal is used by the deadtime logic. 10 OUT45_0 bit is used by the deadtime logic. 11 EXTB[0] signal is used by the deadtime logic.

42.4.29 Master Control Register (FlexPWM_MCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 148h offset = 148h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	IPOL				RUN				0				LDOK			
Write									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_MCTRL field descriptions

Field	Description
0–3 IPOL	Current Polarity This buffered read/write bit is used to select between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output in each submodule. IPOL is ignored in independent mode.

Table continues on the next page...

FlexPWM_MCTRL field descriptions (continued)

Field	Description
	IPOL bit / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3 **NOTE:** The IPOL bit does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading the IPOL bit reads the buffered value and not necessarily the value currently in effect. 0 PWM23 (PWM submodule) is used to generate complementary PWM pair. 1 PWM45 (PWM submodule) is used to generate complementary PWM pair.
4–7 RUN	Run This read/write bit enables the clocks to the PWM generator in each submodule. When RUN equals zero, the submodule counter is reset. A reset clears RUN. **RUN bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3 **NOTE:** For proper initialization of the LDOK and RUN bits, see Initialization . 0 PWM generator disabled 1 PWM generator enabled
8–11 CLDOK	Clear Load Okay This write only bit is used to clear the LDOK bit. Write a 1 to this location to clear the corresponding LDOK. If a reload occurs with LDOK set at the same time that CLDOK is written, then the reload will not be performed and LDOK will be cleared. This bit is self clearing and always reads as a 0. **CLDOK bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3
12–15 LDOK	Load Okay This field loads the PRSC bits of SUB*n*_CTRL1, SUB*n*_INIT, and SUB*n*_VAL*x* registers into a set of buffers in each submodule. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if LDMOD is clear or immediately if LDMOD is set. Set LDOK by reading it when it is logic zero and then writing a logic one to it. SUB*n*_VAL*x*, SUB*n*_INIT, and

Table continues on the next page...

FlexPWM_MCTRL field descriptions (continued)

Field	Description
	SUB*n*_CTRL*x*[PRSC] cannot be written while LDOK is set. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a logic 1 to CLDOK. This bit cannot be written with a zero. LDOK can be set in DMA mode when the DMA indicates that it has completed the update of all SUB*n*_INIT, SUB*n*_VAL*x*, or SUB*n*_CTRL*x*[PRSC]> registers. Reset clears LDOK. **LDOK bit** / **PWM submodule** 0 / 0 1 / 1 2 / 2 3 / 3 **NOTE:** For proper initialization of the LDOK and RUN bits, see Initialization . 0 Do not load new values. 1 Load prescaler, modulus, and PWM values.

42.4.30 Fault Control Register (FlexPWM_FCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 14Ch offset = 14Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_FCTRL field descriptions

Field	Description
0–3 FLVL	Fault Level These read/write bits select the active logic level of the individual fault inputs. A reset clears FLVL. **FLVL bit** / **Fault input** 0 / 0 1 / 1 2 / 2 3 / 3

Table continues on the next page...

FlexPWM_FCTRL field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>0 A logic 0 on the fault input indicates a fault condition.
1 A logic 1 on the fault input indicates a fault condition.</td></tr>
<tr><td>4–7
FAUTO</td><td>Automatic Fault Clearing
These read/write bits select automatic or manual clearing of faults. A reset clears FAUTO.
<table><tr><th>FAUTO bit</th><th>Fault input</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>3</td></tr></table>
0 Manual fault clearing. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending the state of the FFULL bits. This is further controlled by the FSAFE bits.
1 Automatic fault clearing. PWM outputs disabled by this fault are enabled when the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of FFLAGx bit.</td></tr>
<tr><td>8–11
FSAFE</td><td>Fail Safe Mode
These read/write bits select the safety mode during manual fault clearing. A reset clears FSAFE.
<table><tr><th>FSAFE bit</th><th>Fault input</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>3</td></tr></table>
NOTE: The FFPINx bit may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency.
0 Normal mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of the FFPINx bit. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts. This is because the fault inputs have a direct combinational link to disable the PWM outputs (as programmed in DISMAP).
1 Safe mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear and the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits.</td></tr>
<tr><td>12–15
FIE</td><td>Fault Interrupt Enables
This read/write bit enables CPU interrupt requests generated by the FAULTx pins. A reset clears FIE.
<table><tr><th>FIE bit</th><th>Fault input</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
</table>

Table continues on the next page...

FlexPWM_FCTRL field descriptions (continued)

Field	Description
	<table><tr><th>FIE bit</th><th>Fault input</th></tr><tr><td>3</td><td>3</td></tr></table> **NOTE:** The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0 FAULT*x* CPU interrupt requests disabled. 1 FAULT*x* CPU interrupt requests enabled.

42.4.31 Fault Status Register (FlexPWM_FSTS)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 14Eh offset = 14Eh

Bit	0	1	2	3	4	5	6	7
Read	0			FTEST	FFPIN			
Write								
Reset	0	0	0	0	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read	FFULL				FFLAG			
Write					w1c			
Reset	0	0	0	0	*	*	*	*

* Notes:
- FFLAG field: Undefined
- FFPIN field: Undefined

FlexPWM_FSTS field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FTEST	Fault Test These read/write bit is used to simulate a fault condition. Setting this bit will cause a simulated fault to be sent into all of the fault filters. The condition will propagate to the fault flags and possibly the PWM outputs depending on the DISMAP settings. Clearing this bit removes the simulated fault condition.

Table continues on the next page...

FlexPWM_FSTS field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>0 No fault.
1 Cause a simulated fault.</td></tr>
<tr><td>4–7
FFPIN</td><td>Filtered Fault Pins

These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on FFPIN.

<table>
<tr><th>FFPIN bit</th><th>Fault input</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table></td></tr>
<tr><td>8–11
FFULL</td><td>Full Cycle

These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition.

<table>
<tr><th>FFULL bit</th><th>Fault input</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWM outputs are re-enabled at the start of a full or half cycle.
1 PWM outputs are re-enabled only at the start of a full cycle.</td></tr>
<tr><td>12–15
FFLAG</td><td>Fault Flags

These read-only flags are set in either the fourth or fifth cycle (depending on clock synchronisation) of the FlexPWM clock, after a transition to active on the FAULTx pin. Clear FFLAGx by writing a logic one to it. A reset clears FFLAG.

<table>
<tr><th>FFLAG bit</th><th>Fault input</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 No fault on the FAULTx pin.
1 Fault on the FAULTx pin.</td></tr>
</table>

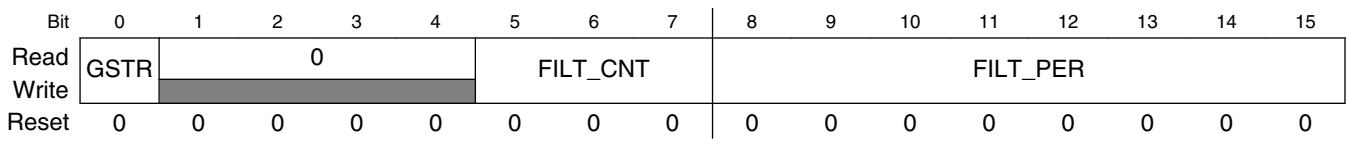
42.4.32 Fault Filter Register (FlexPWM_FFILT)

The settings in this register are shared among each of the fault input filters.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 150h offset = 150h

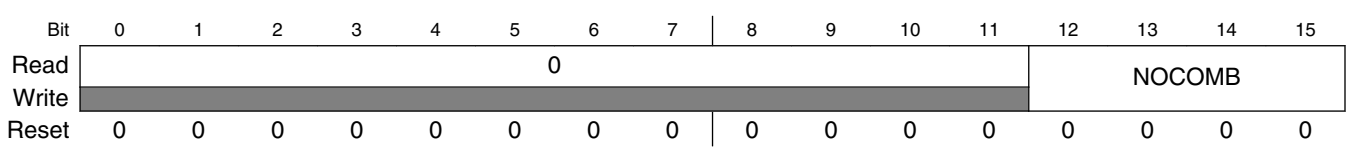


FlexPWM_FFILT field descriptions

Field	Description
0 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 peripheral clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0 Fault input glitch stretching is disabled. 1 Input fault signals will be stretched to at least 2 peripheral clock cycles.
1–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in Input Filter Considerations .
8–15 FILT_PER	Fault Filter Period These bits represent the sampling period (in peripheral clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in Input Filter Considerations .

42.4.33 Fault Control 2 Register (FlexPWM_FCTRL2)

Address: 0h base + 152h offset = 152h



FlexPWM_FCTRL2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 NOCOMB	No combinational path from fault inputs to PWM outputs This read/write field is used to control the combinational path from the fault inputs to the PWM outputs. When these bits are low (default), the corresponding fault inputs have a combinational path to the PWM outputs that are sensitive to these fault inputs (as defined by DISMAP). This combinational path is a safety feature that ensures the output is disabled even if the SOC has a failure of its clocking system. The combinational path also means that a pulse on the fault input can cause a brief disable of the PWM output even if the fault pulse is not wide enough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filtered and latched fault signals as the fault source to disable the PWM outputs. This eliminates fault glitches from creating PWM output glitches but also increases the latency to respond to a real fault. 0 There is a combinational link from the fault inputs to the PWM outputs. The fault inputs are combined with the filtered and latched fault signals to disable the PWM outputs. 1 The direct combinational path from the fault inputs to the PWM outputs is disabled and the filtered and latched fault signals are used to disable the PWM outputs.

42.4.34 Input Filter Considerations

The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x peripheral clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set the FFLAG and FFPIN bits of the FSTS register.

42.5 Functional Description

This section describes the functionality of the FlexPWM.

42.5.1 PWM Capabilities

This section describes some capabilities of the PWM module.

42.5.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in following figure.

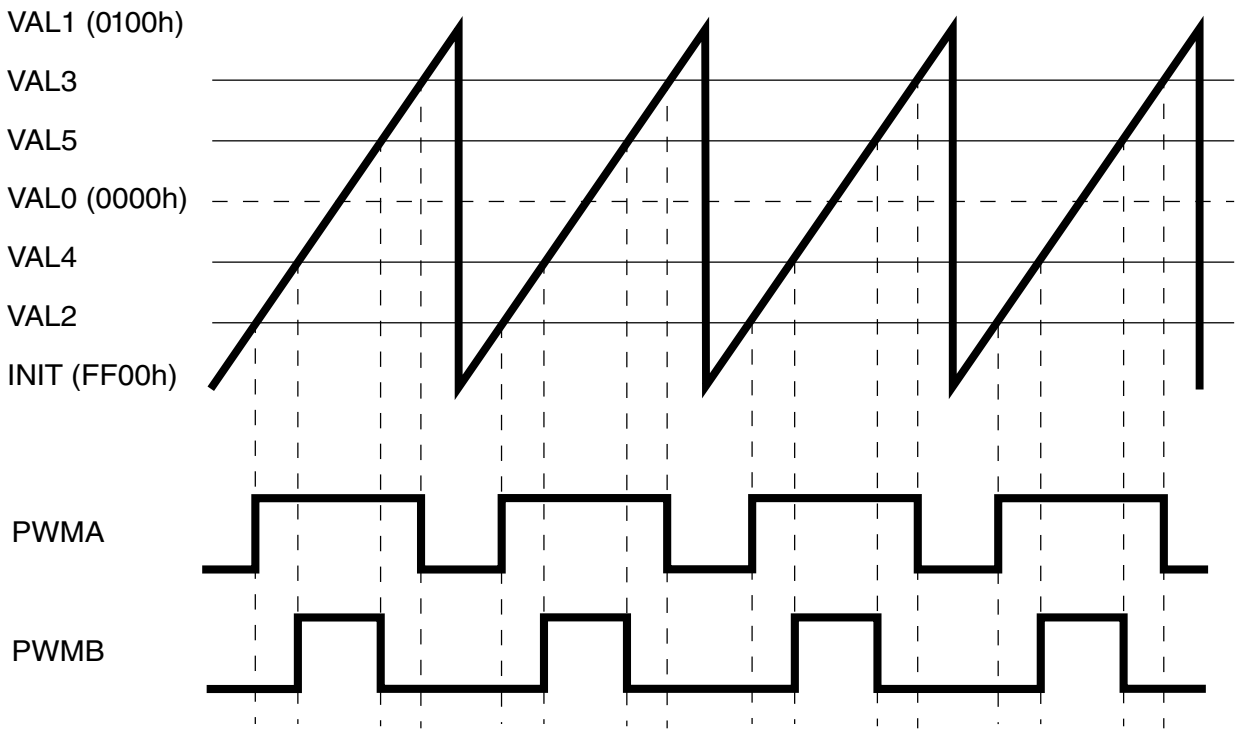


Figure 42-3. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

The figure above also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in

their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

42.5.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as illustrated in the following figure. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

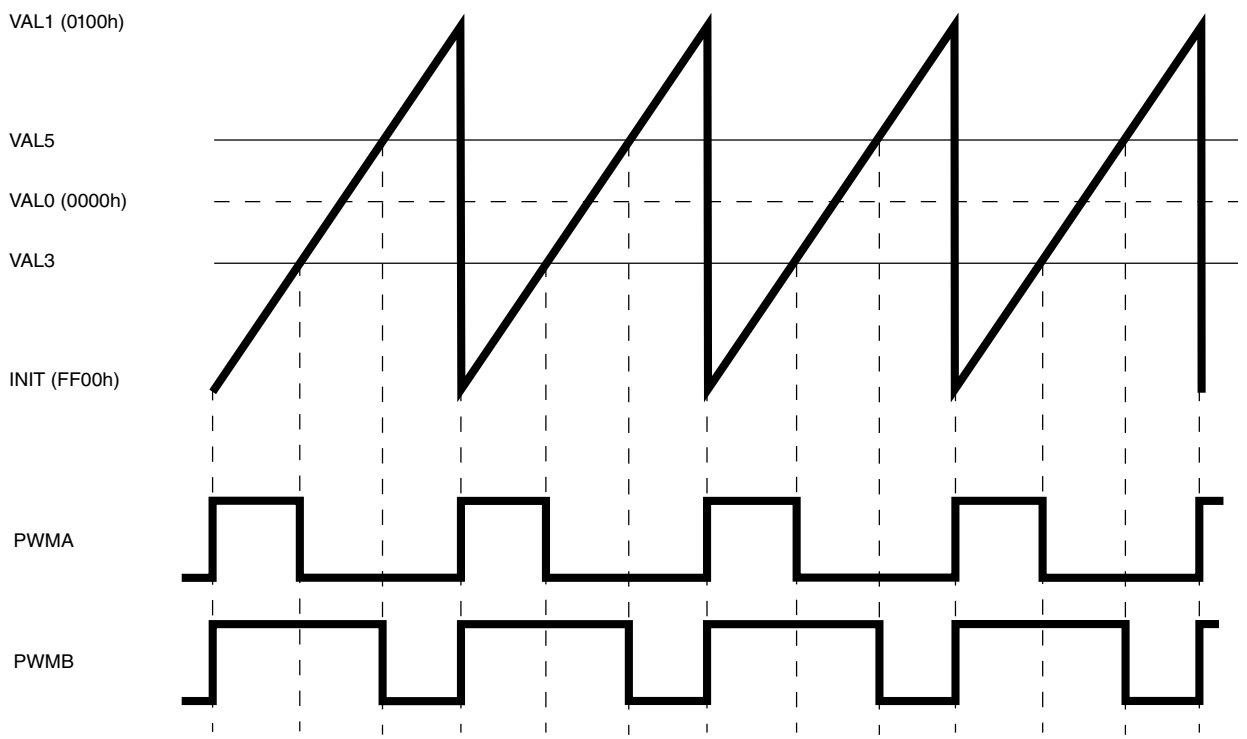


Figure 42-4. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn

off edge value and the motor voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

42.5.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases ARE applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as illustrated in the following figure. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

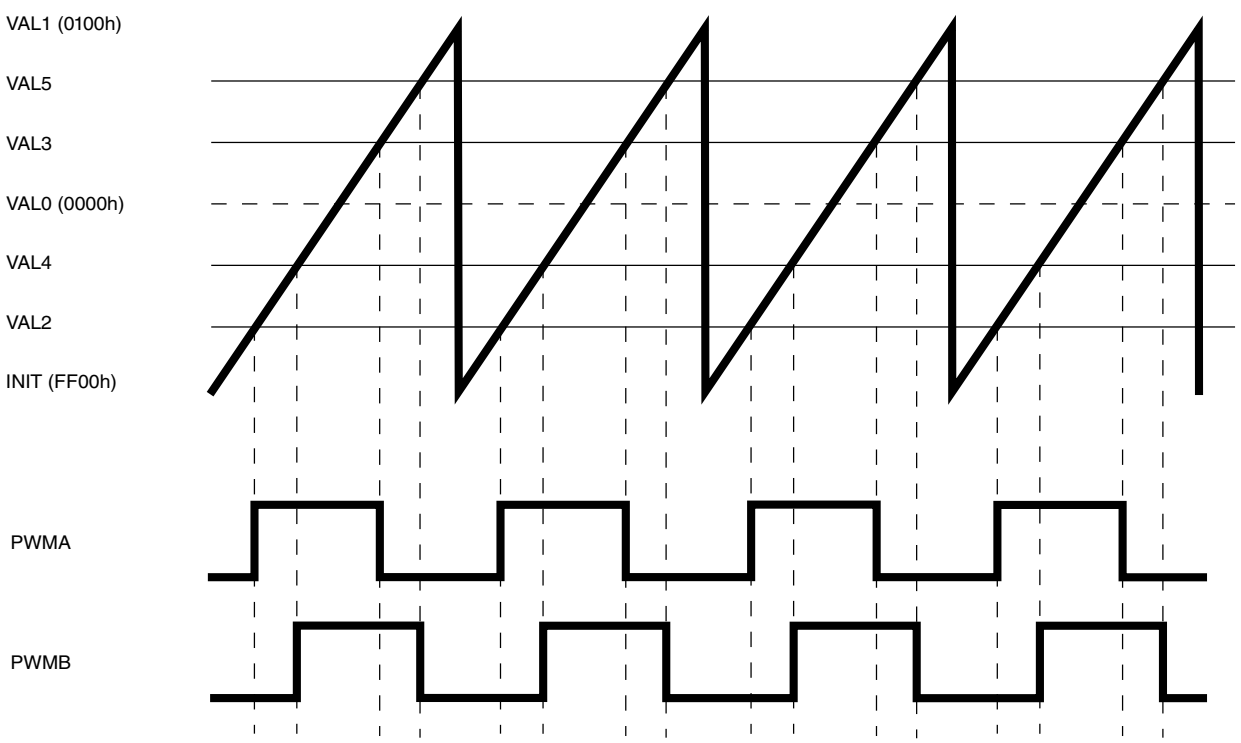


Figure 42-5. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs can be seen in the following figure. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

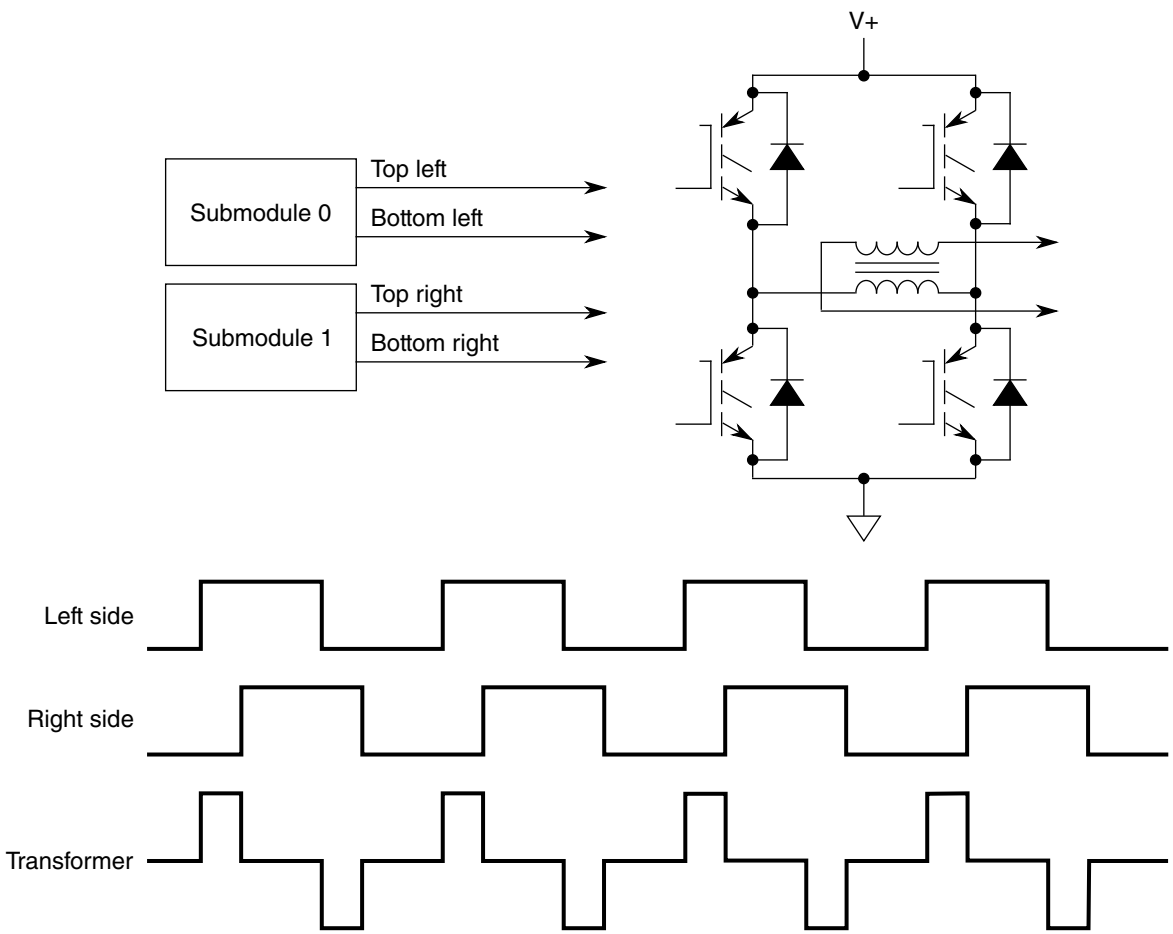


Figure 42-6. Phase Shifted PWMs Applied to a Transformer Primary

42.5.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWMA in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (see Figure 42-17) as shown in the following figure. The DBLPWM signal can be run through the deadtime insertion logic.

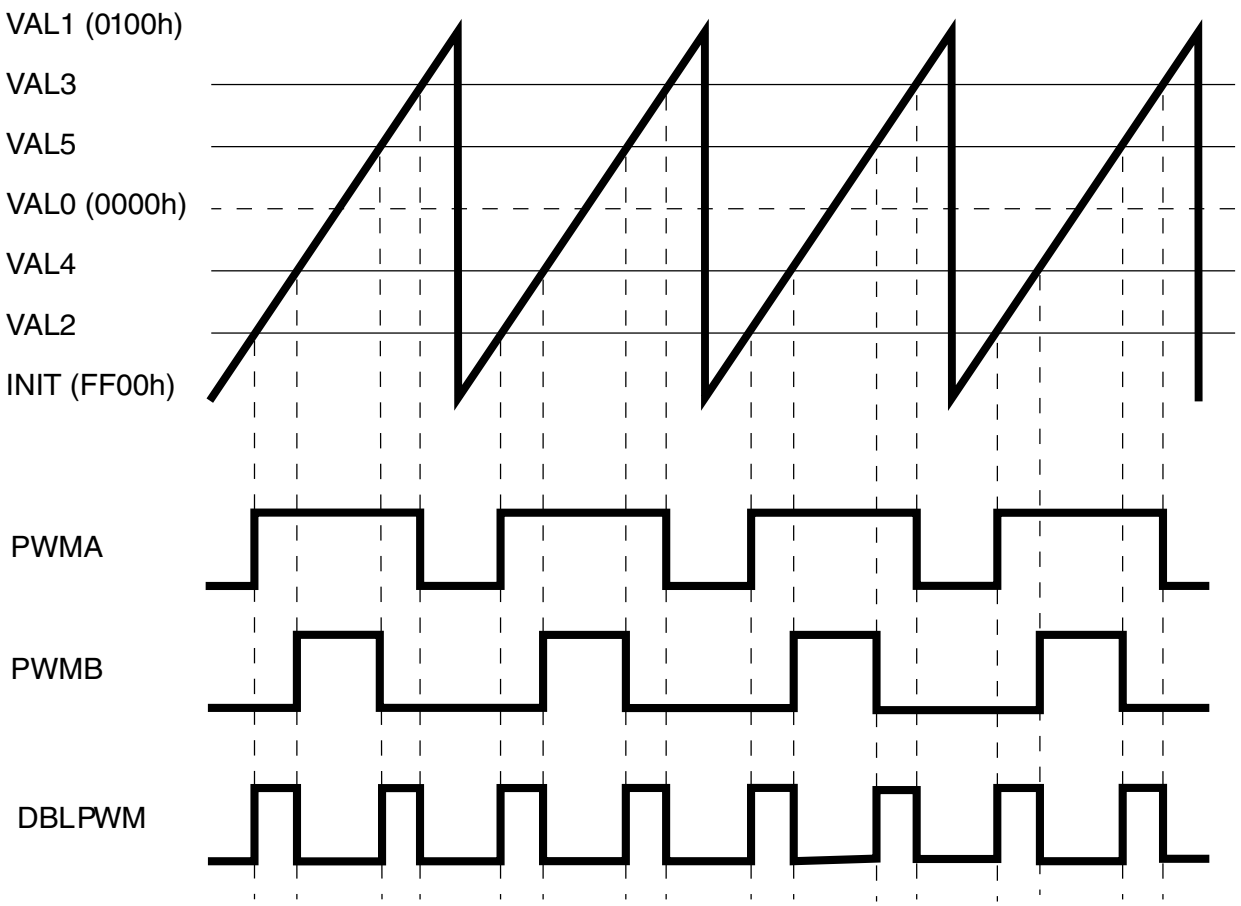


Figure 42-7. Double Switching Output Example

42.5.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. the following figure shows how this is accomplished. When specifying complimentary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators

are free to perform other functions. In this example, the software doesn't have to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

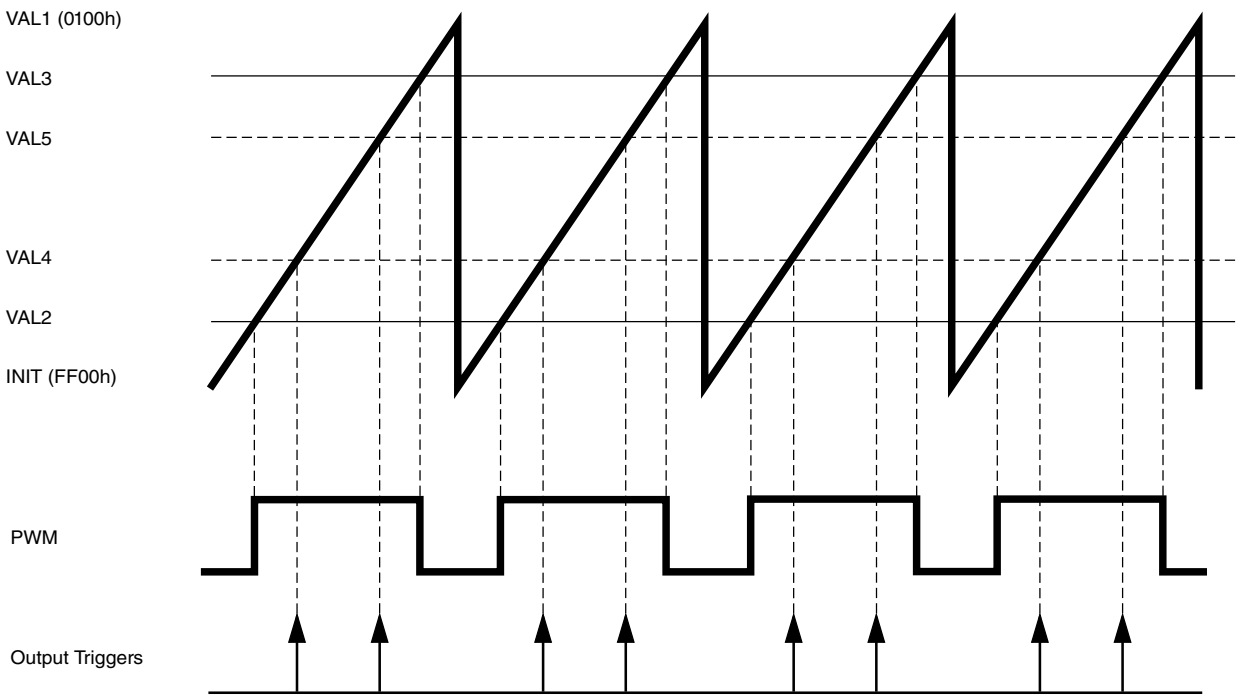


Figure 42-8. Multiple Output Trigger Generation in Hardware

Since each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. The following figure shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In this figure, ALL submodule comparators are shown being used for ADC trigger generation.

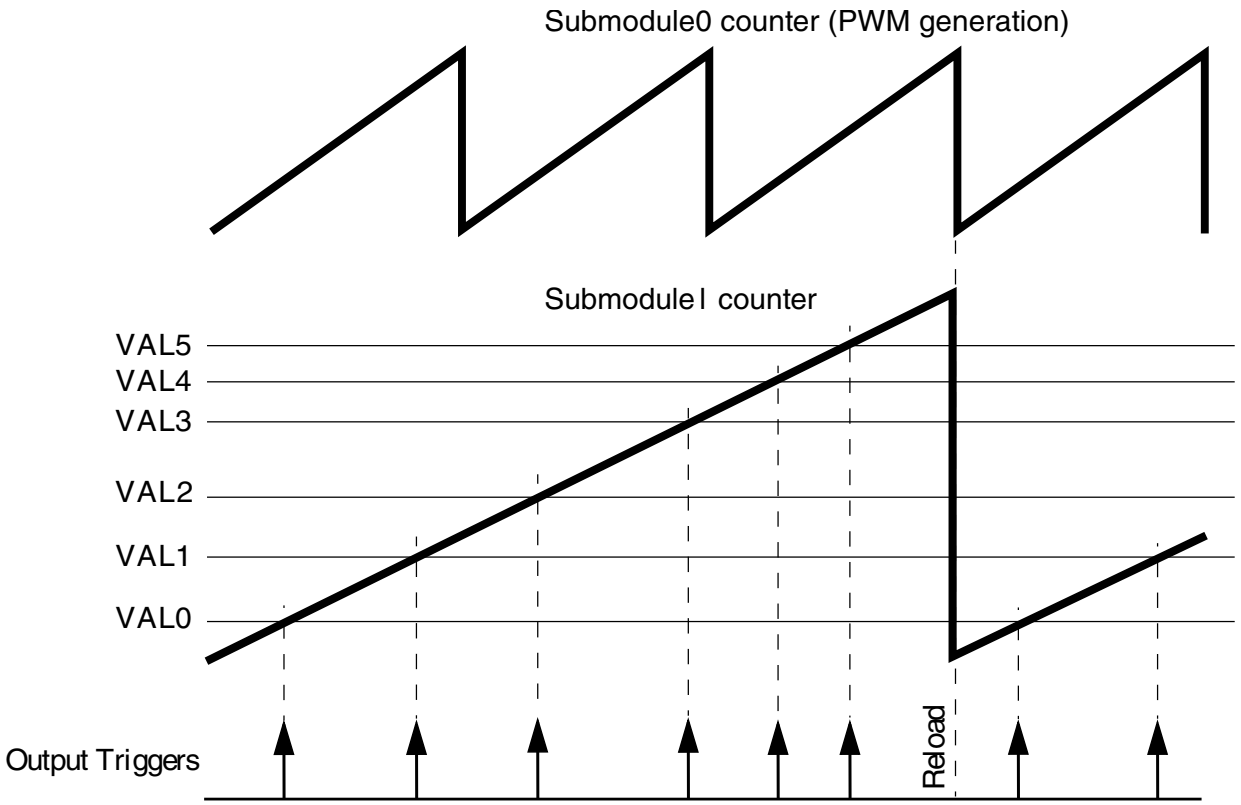


Figure 42-9. Multiple Output Triggers Over Several PWM Cycles

42.5.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWMX pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

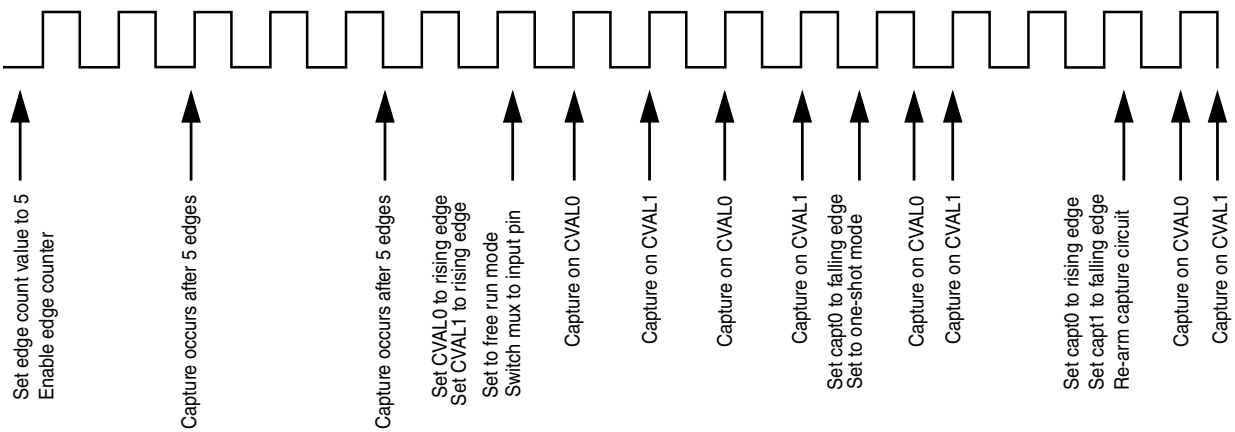


Figure 42-10. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (e.g, PWMX) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. As an example, consider the following figure. In this application the output of a PWM power stage is connected to the PWMX pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays.

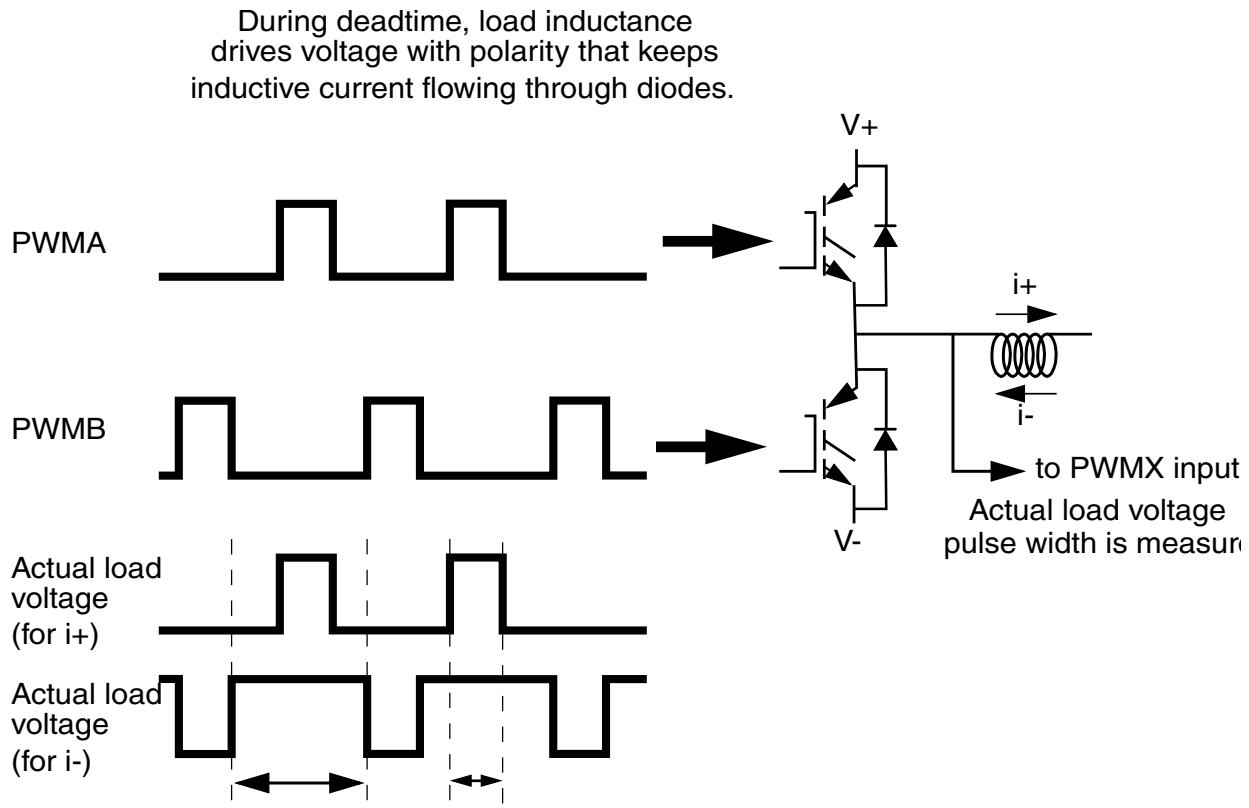


Figure 42-11. Output Pulse Width Measurement Possible with the E-Capture Circuit

42.5.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event using the SWCOUT and DTSRCSEL registers. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

The following figure shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of the following figure are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

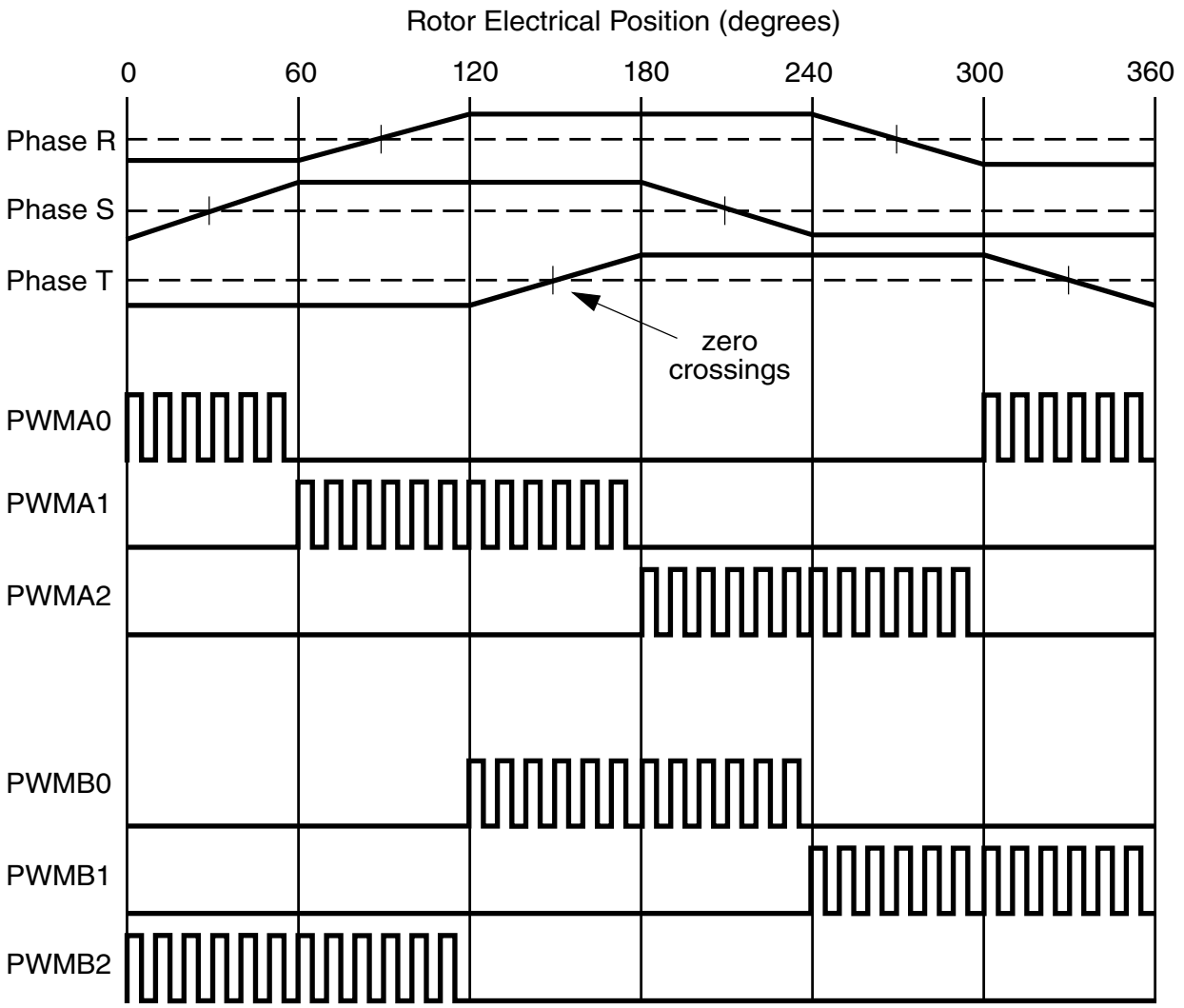


Figure 42-12. Sensorless BLDC Commutation Using the Force Out Function

42.5.1.8 Phase shifted full bridge for DC/DC converters

Use CTRL1[PHSSHFT] to allow the EXT_SWITCH input to control the output of PWMA0/PWMB0. The EXT_SWITCH input switches the source of the submodule0 output from the SUB0_VAL*x* registers to the SUB1_VAL*x* registers. This muxing can be seen in Figure 42-16.

42.5.2 Functional Details

This section describes the implementation of various features of the PWM in greater detail.

42.5.2.1 PWM Clocking

The following figure shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the peripheral clock, EXT_CLK, and AUX_CLK. The EXT_CLK is generated by an on-chip resource such as a Timer module and goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8 bit prescaler from submodule0 can control all of the submodules.

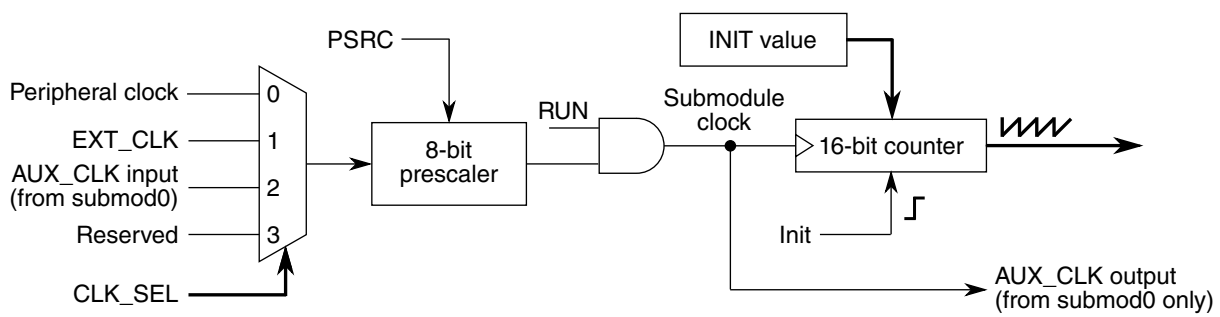


Figure 42-13. Clocking Block Diagram for Each PWM Submodule

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the peripheral clock frequency by 1-128. The prescaler bits, PRSC, in the control register (CTRL1), select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins or LDMOD is set.

42.5.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using the LDFQ bits and the FULL bit. A half cycle reload option is also supported (HALF) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the SUB*n*VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in the following figure, the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

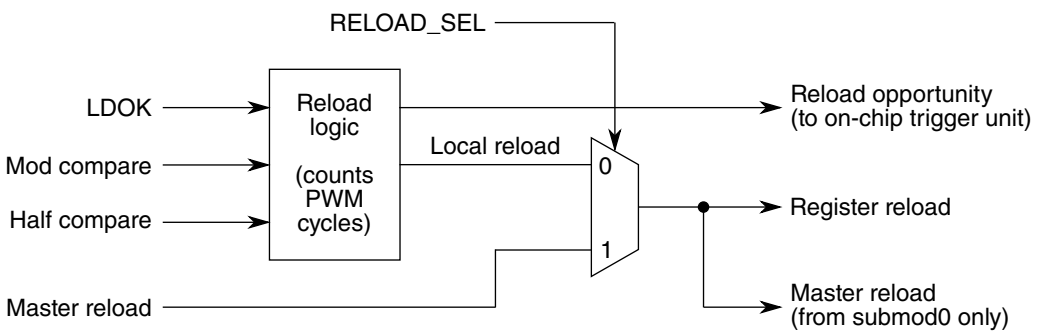


Figure 42-14. Register Reload Logic

42.5.2.3 Counter Synchronization

Referring to the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

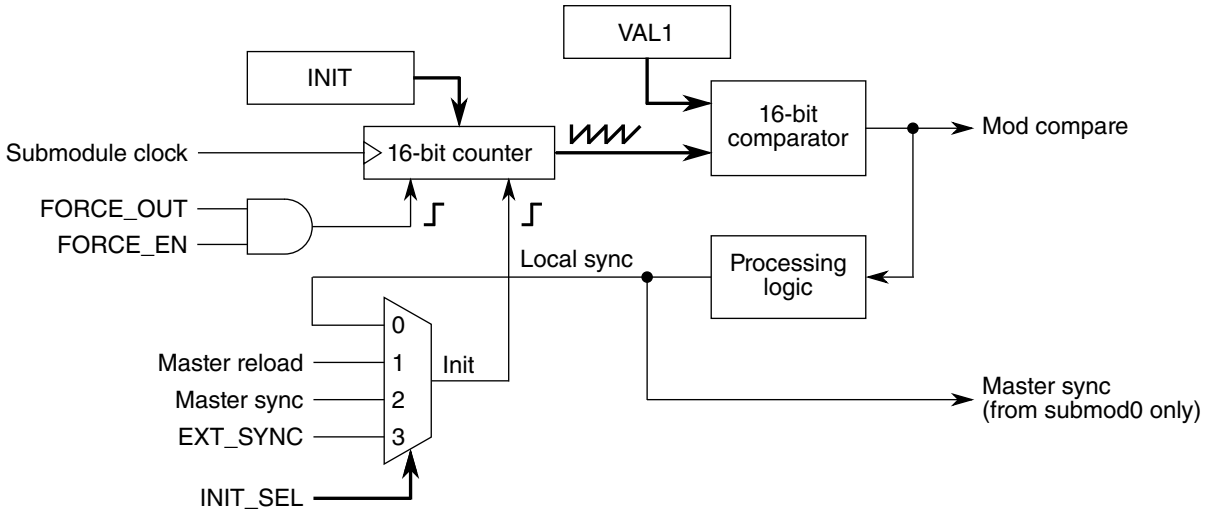


Figure 42-15. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0. The SUB*n*_VAL1 register and associated comparator of the other submodules can then be freed up for other functions such as PWM generation, input captures, output compares, or output triggers.

The EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that the FORCE_EN bit is set. As indicated by the figure above, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of

only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

42.5.2.4 PWM Generation

The following figure illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated SUB*n*_VAL*x* registers are utilized for each PWM output signal. One comparator and VALx register is used to control the turn on edge while a second comparator and SUB*n*_VAL*x* register control the turn off edge.

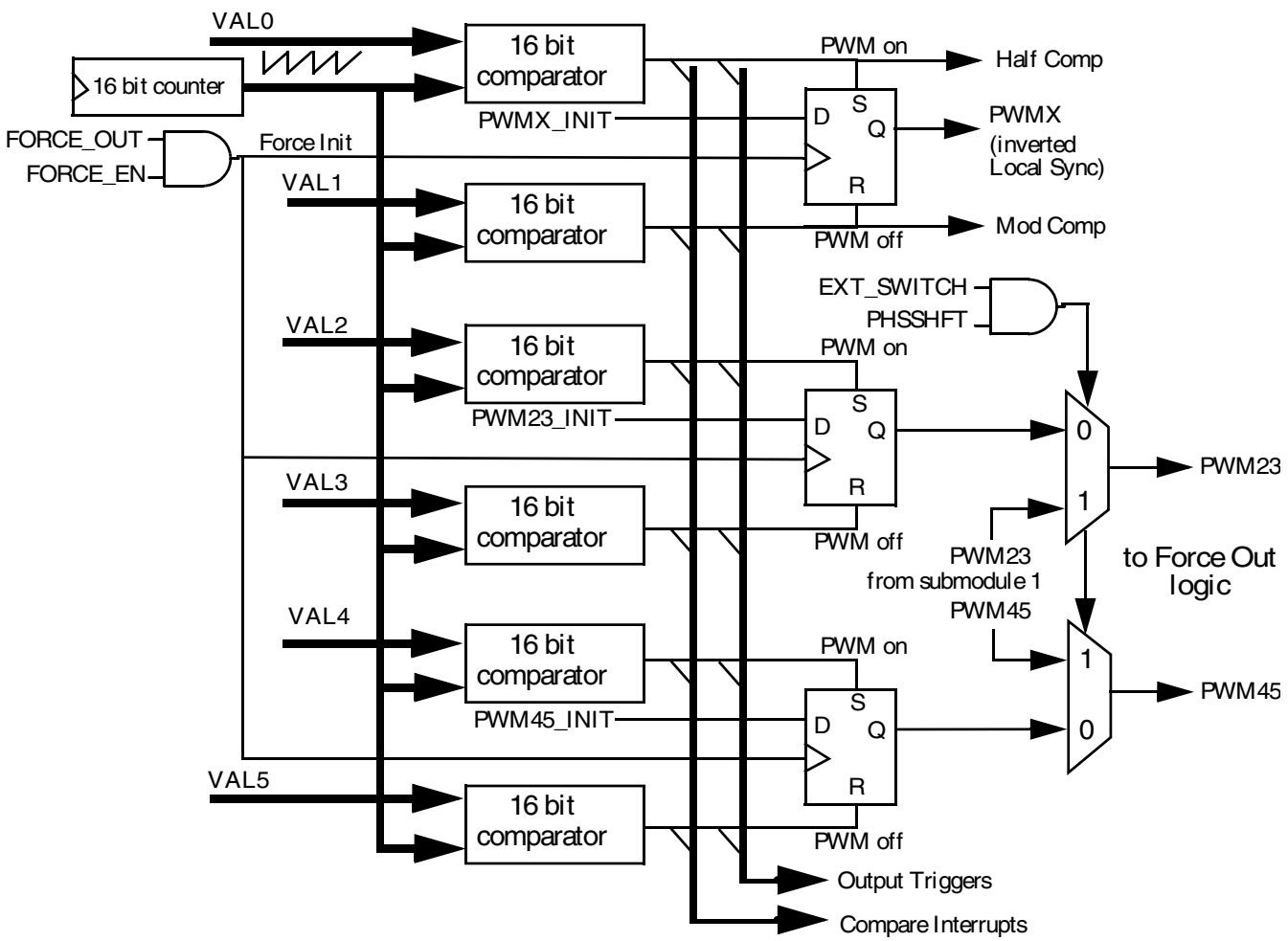


Figure 42-16. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a rising edge of the Local Sync signal, comparator 1 generates a falling edge. Comparator 1 is also hardwired to the reload logic to generate the half cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the SUB*n*_VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the SUB*n*_VAL1 and SUB*n*_VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal effectively turning it into an auxiliary PWM signal (PWMX) assuming that the PWMX pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating 3 PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The muxes for PWM23 and PWM45, shown in the figure above, are only in submodule0.

If the comparators assert both the set and reset of the flip-flop at the same time, then the flop output goes to 0.

42.5.2.5 Output Compare Capabilities

By using the SUB*n*_VAL*x* registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

Referring again to Figure 42-16, an output compare is initiated by programming a SUB*n*_VAL*x* register for a timer compare which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWMA signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the SUB*n*_VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWMA signal that sets it low, the SUB*n*_VAL3 register is programmed with the appropriate count value and the SUB*n*_VAL2 register is programmed with a

value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

42.5.2.6 Force Out Logic

For each submodule software can select between seven signal sources for the FORCE_OUT signal: the local FORCE bit, the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, or the EXT_FORCE signal from on or off chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master signals or EXT_FORCE signal should be selected.

The following figure illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the EXTA or EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux which immediately switches the requested signal to the output of the mux for further processing downstream.

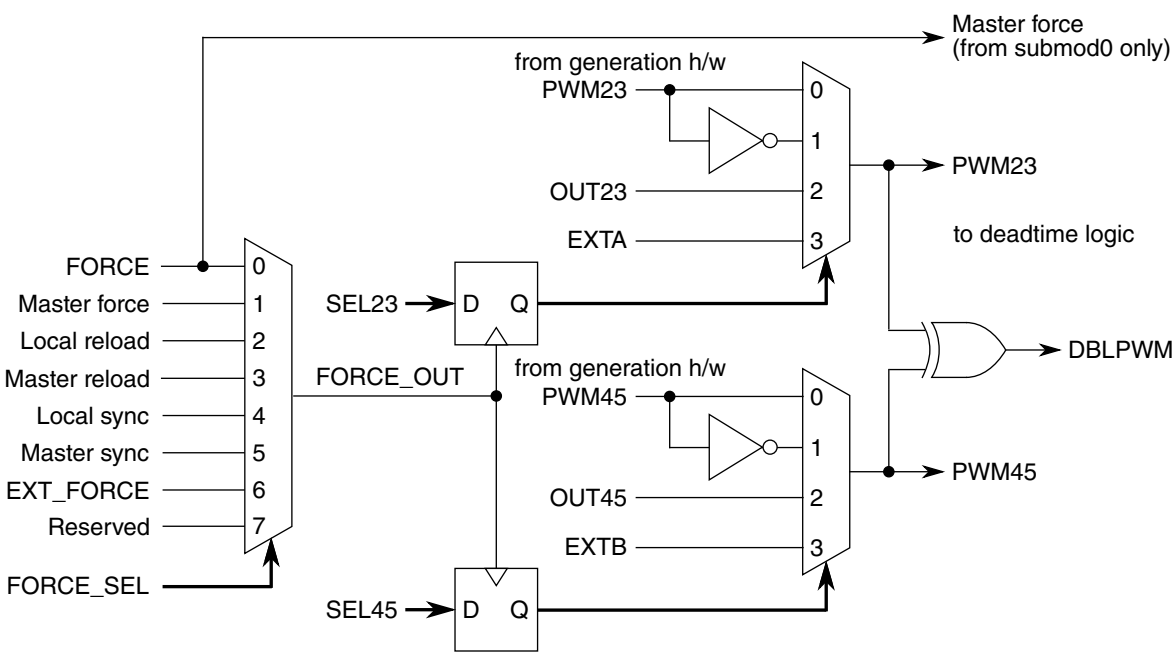


Figure 42-17. Force Out Logic

The local FORCE signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the local FORCE bit of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

42.5.2.7 Independent or Complementary Channel Operation

Writing a logic one to the INDEP bit of the CTRL2 register configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to the INDEP bit configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in the following figure in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by the IPOL bit.

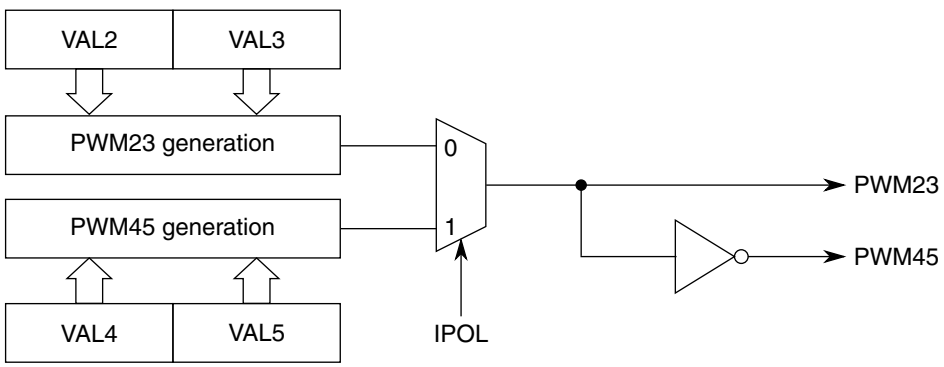


Figure 42-18. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in the following figure.

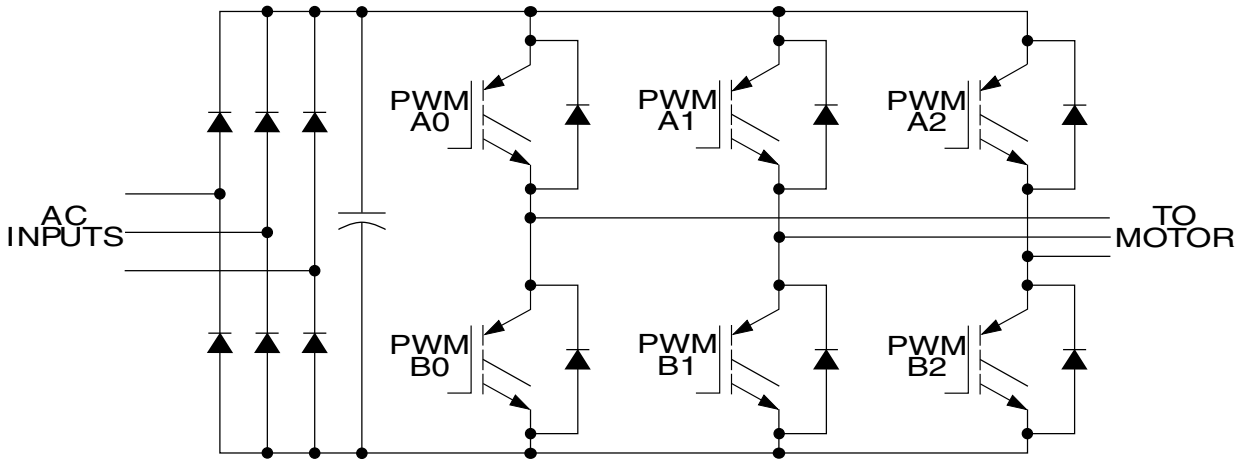


Figure 42-19. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

42.5.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

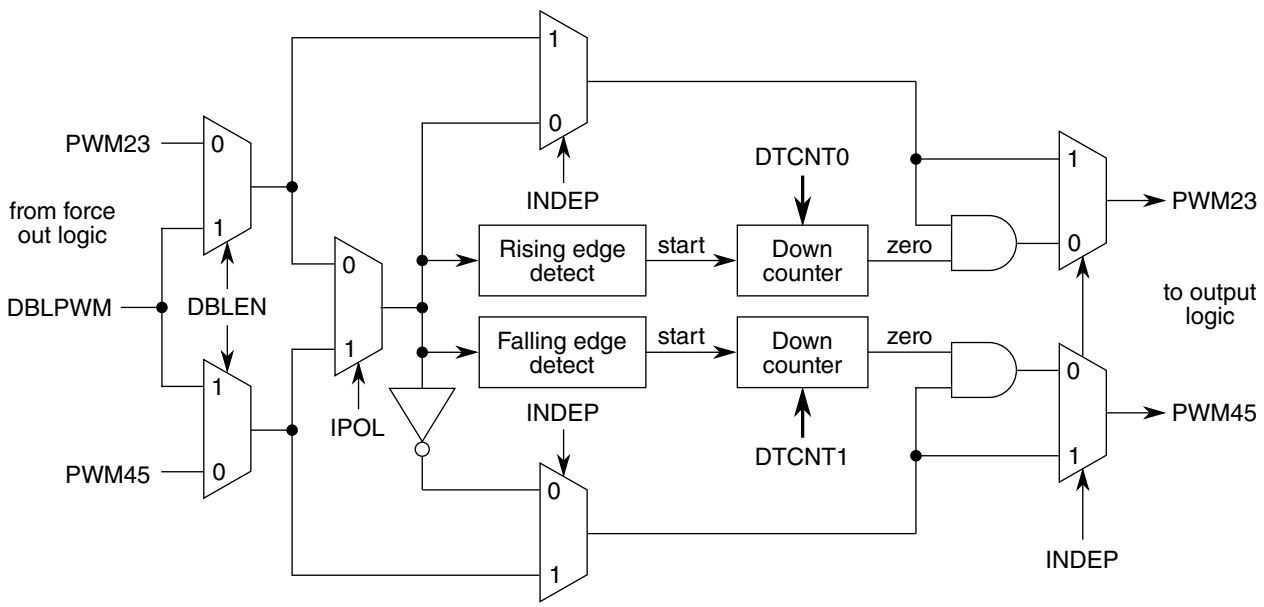


Figure 42-20. Deadtime Insertion and Fine Control Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure above. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. However, the transistor's characteristics may cause its switching-off time to be longer than its switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of peripheral clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

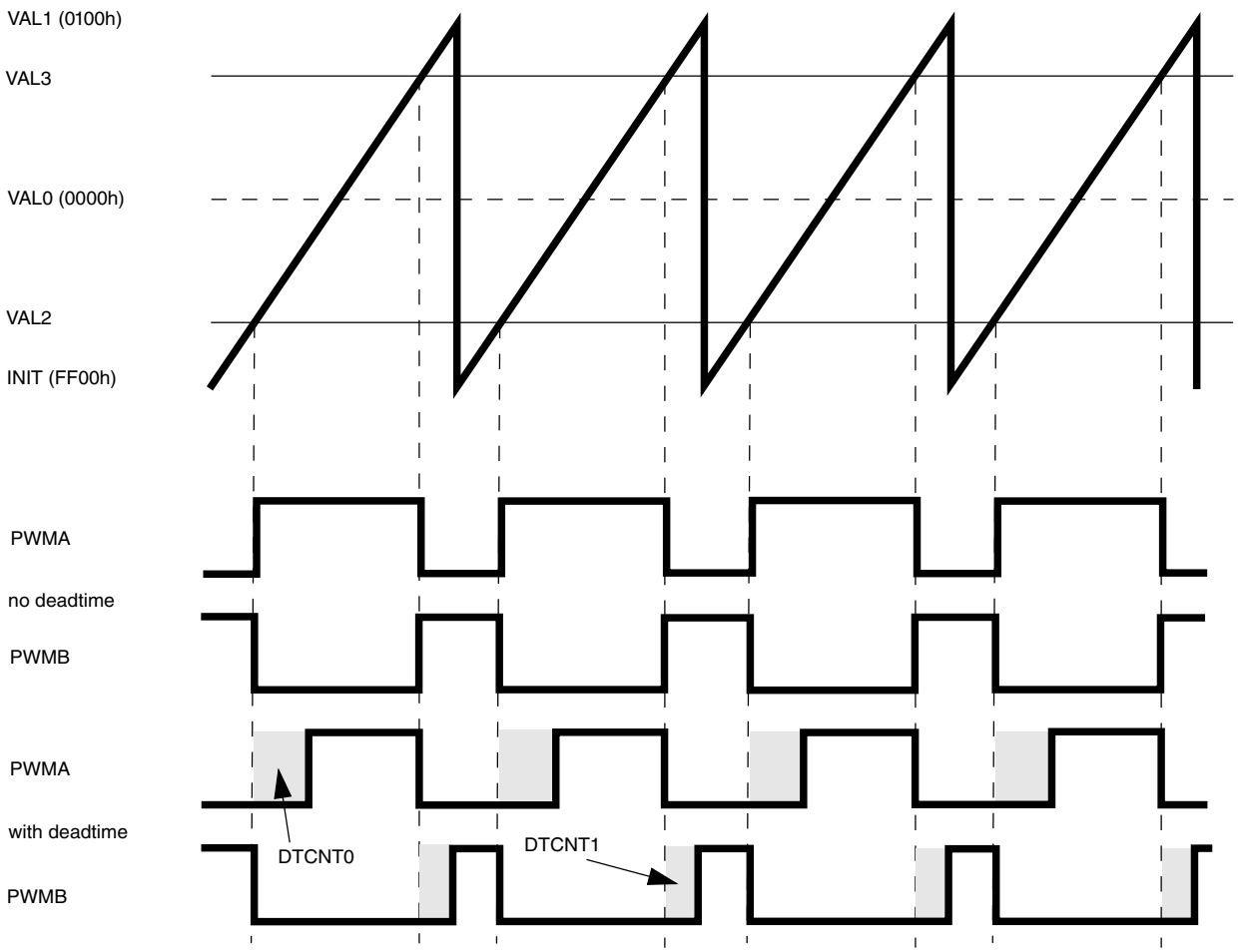


Figure 42-21. Deadtime Insertion

42.5.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

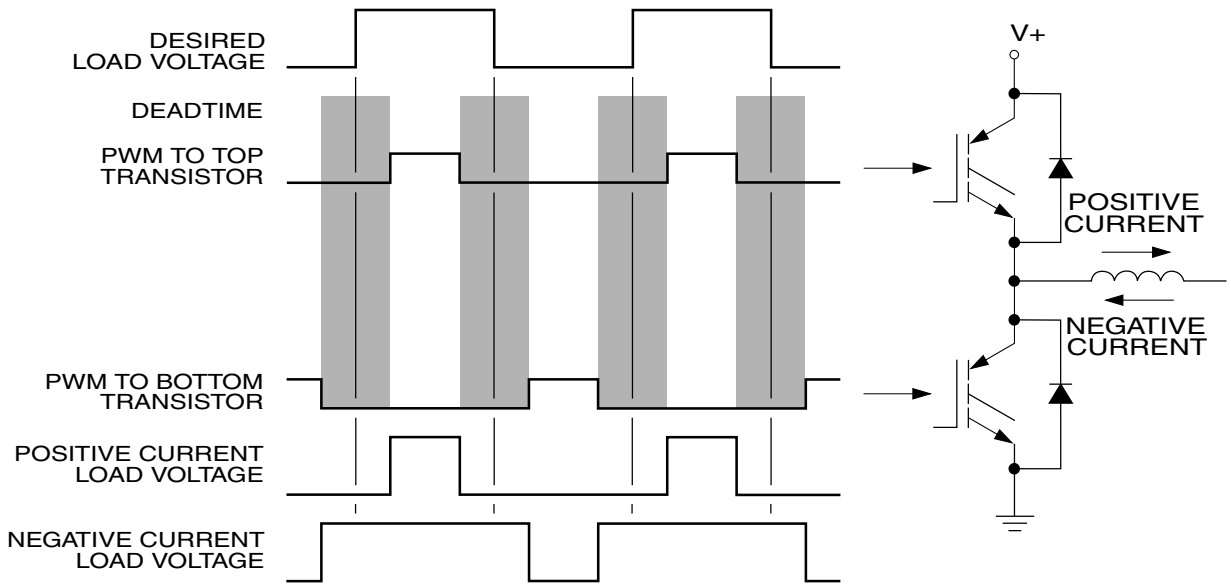


Figure 42-22. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair. See the figure above. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the SUB*n*_VAL*x* registers. Either the SUB*n*_VAL2/SUB*n*_VAL3 or the SUB*n*_VAL4/SUB*n*_VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the SUB*n*_VAL2/SUB*n*_VAL3 or SUB*n*_VAL4/SUB*n*_VAL5 pair is active depends on either:

- The state of the current status pin, PWMx, for that driver
- The state of the odd/even correction bit, IPOL, for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate SUB*n*_VAL*x* register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

42.5.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMx pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in the DTx bits in the CTRL1 register. The DTx bits are a timing marker especially indicating when to toggle between PWM value registers. Software can then set the IPOL bit to switch between SUB*n*_VAL2/SUB*n*_VAL3 and SUB*n*_VAL4/SUB*n*_VAL5 register pairs according to DTx values.

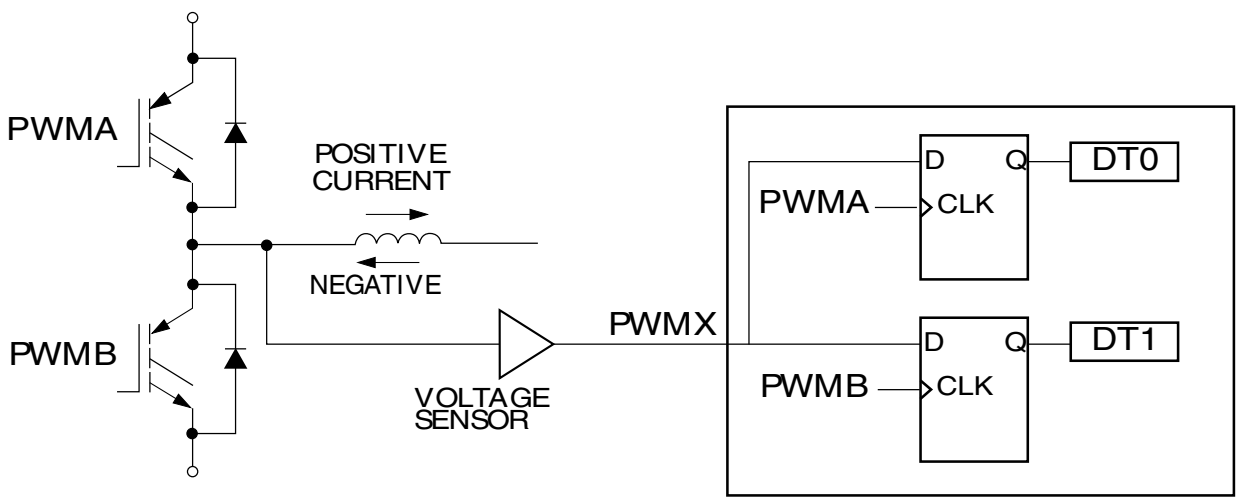


Figure 42-23. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, DT0 = 0, DT1 = 0, during deadtime periods if current is large and flowing out of the complementary circuit. See the figure above. Both D flip-flops latch the high, DT0 = 1, DT1 = 1, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results will be DT0 = 0 and DT1 = 1. Thus, the best time to change one PWM value register to another is just before the current zero crossing.

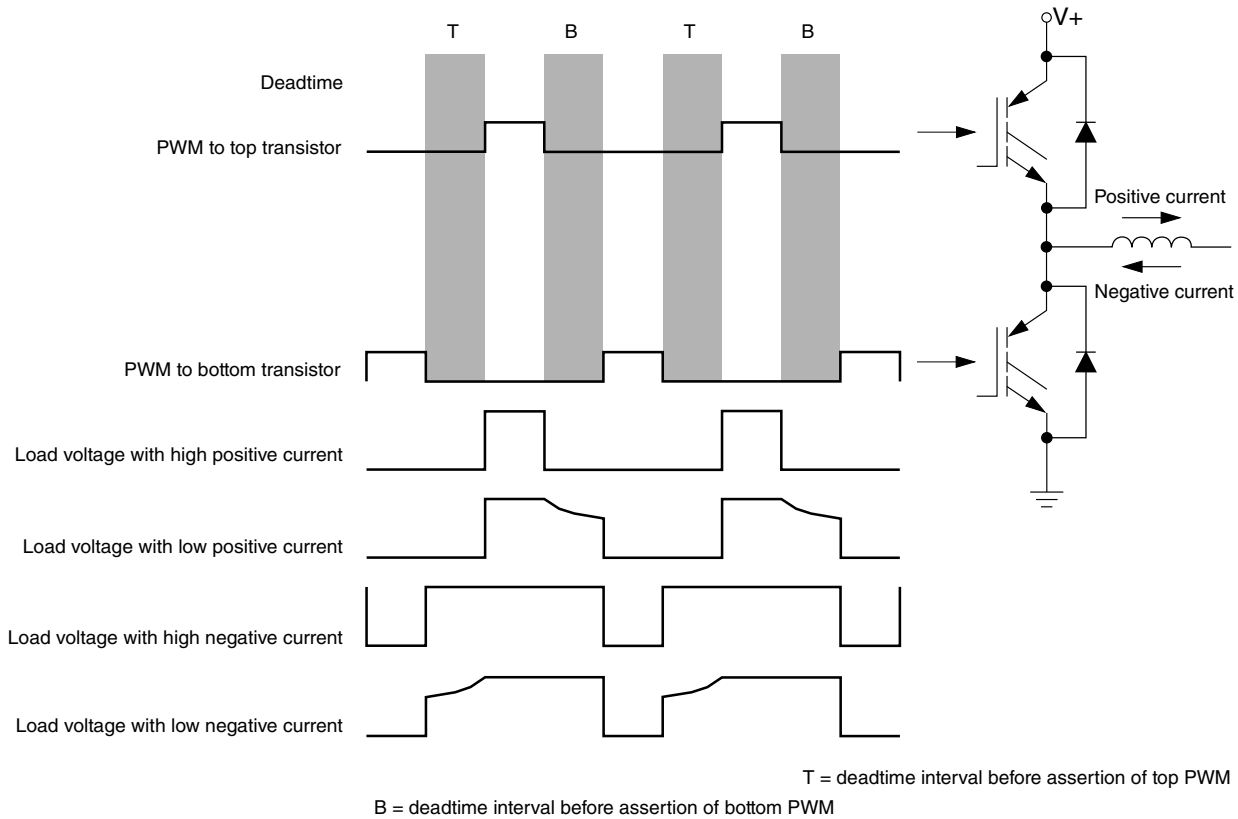


Figure 42-24. Output Voltage Waveforms

42.5.2.9 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic in the figure are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program the

POLA and POLB bits before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the PWMxFS fields.

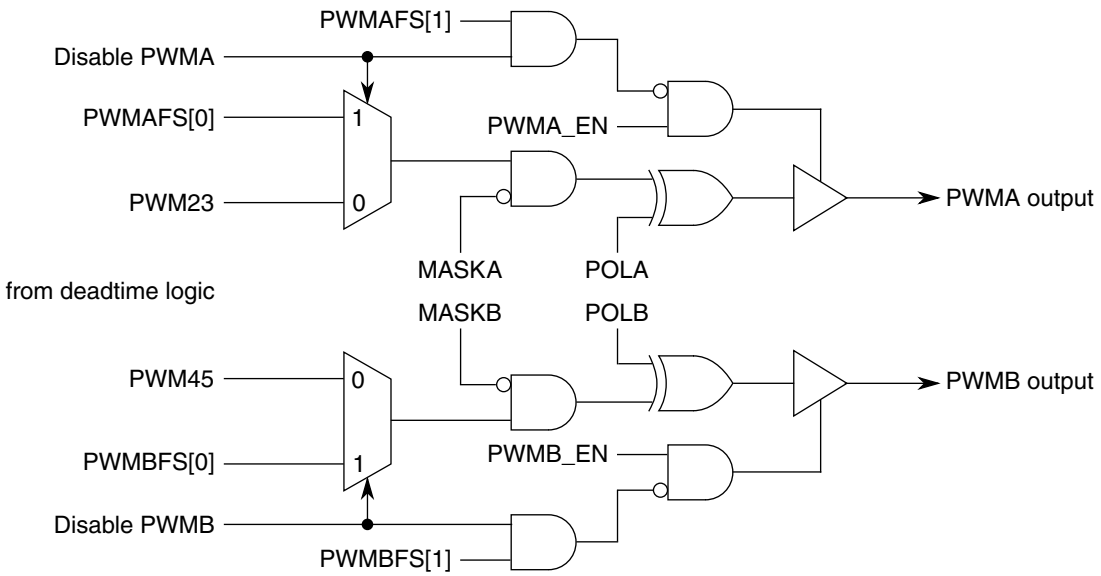


Figure 42-25. Output Logic Section

42.5.2.10 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure illustrates the block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8-bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8-bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

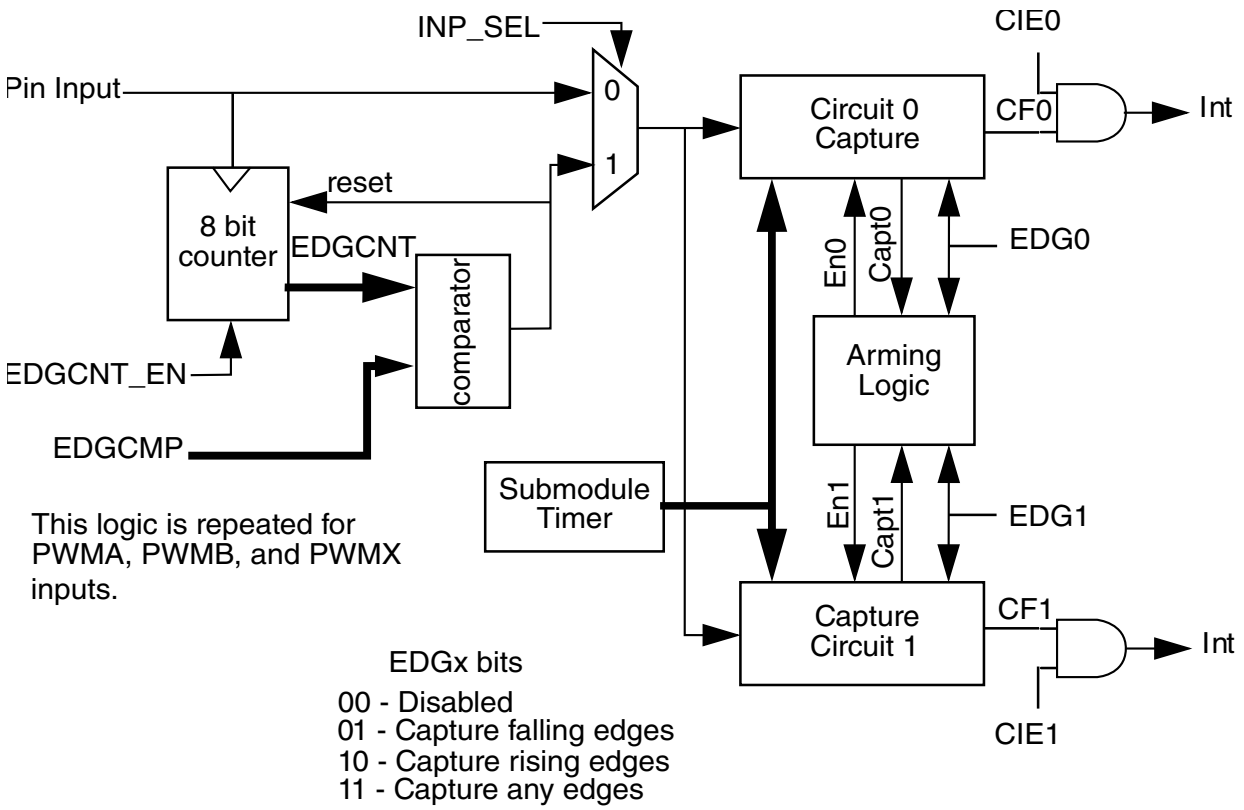


Figure 42-26. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by the EDGx1 and EDGx0 bits whose functionality is listed in the figure above. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

42.5.2.11 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via the FLVL bits. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When

fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of the PWMxFS bits.

The fault decoder disables PWM pins selected by the fault logic and the disable mapping register (DISMAP). See the following figure for an example of the fault disable logic. Each bank of bits in DISMAP control the mapping for a single PWM pin. Refer to following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

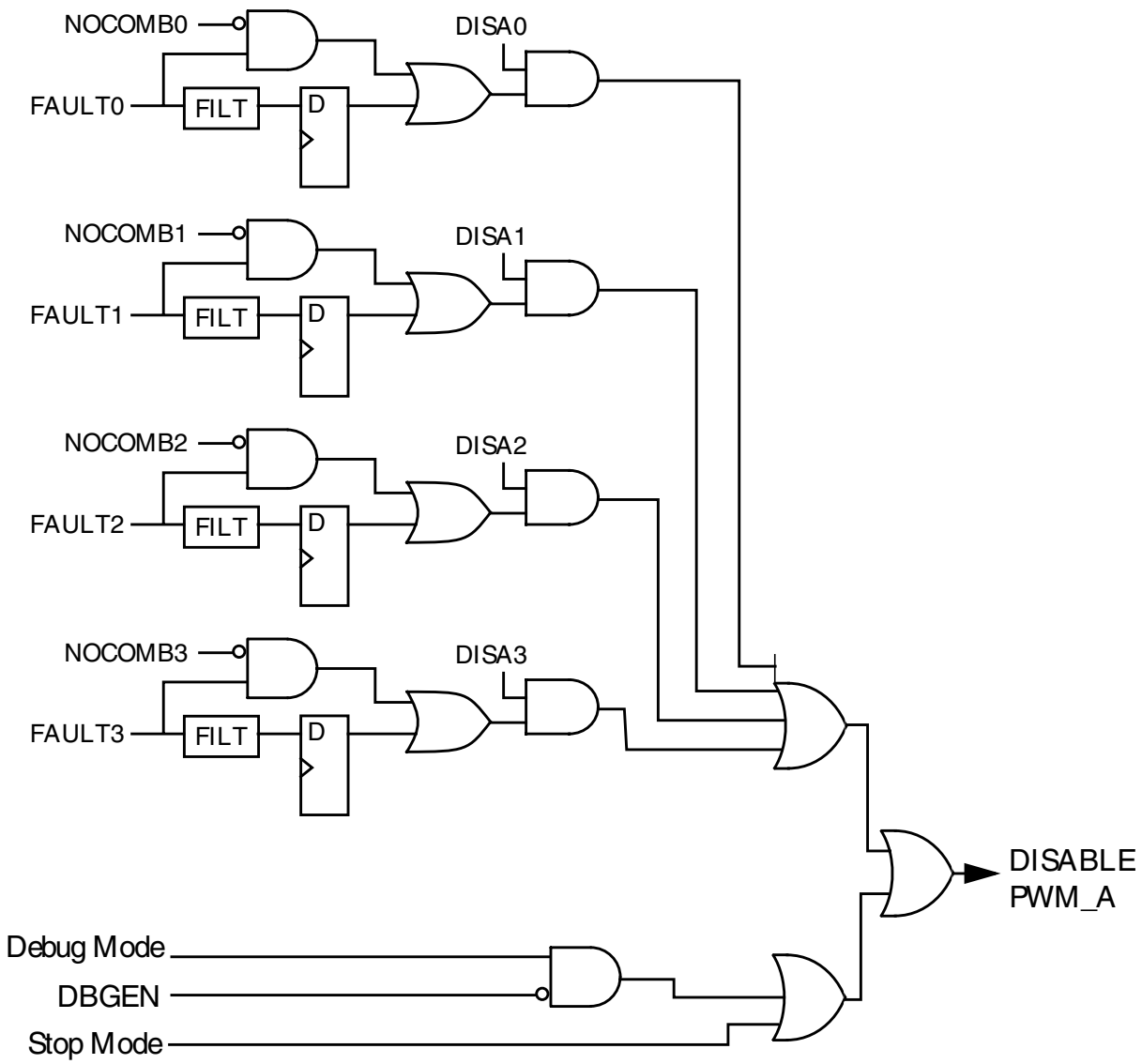


Figure 42-27. Fault decoder for PWMA

Table 42-3. Fault Mapping

PWM Pin	Controlling Register Bits
PWMA	DISA[3:0]
PWMB	DISB[3:0]
PWMX	DISX[3:0]

42.5.2.11.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with the FILT_PER field of the FFILTx register. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using the FILT_CNT field of the same register. Setting FILT_PER to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FLVLx is set), the corresponding FFPINx and fault flag, FFLAGx, bits are set. The FFPINx bit remains set as long as the filtered FAULTx pin is zero. Clear FFLAGx by writing a logic 1 to FFLAGx.

If the FIEx, FAULTx pin interrupt enable bit is set, the FFLAGx flag generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears the FFLAGx flag by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

42.5.2.11.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for automatic clearing.

When FAUTOx is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle. Clearing the FFLAGx flag does not affect disabled PWM pins when FAUTOx is set.

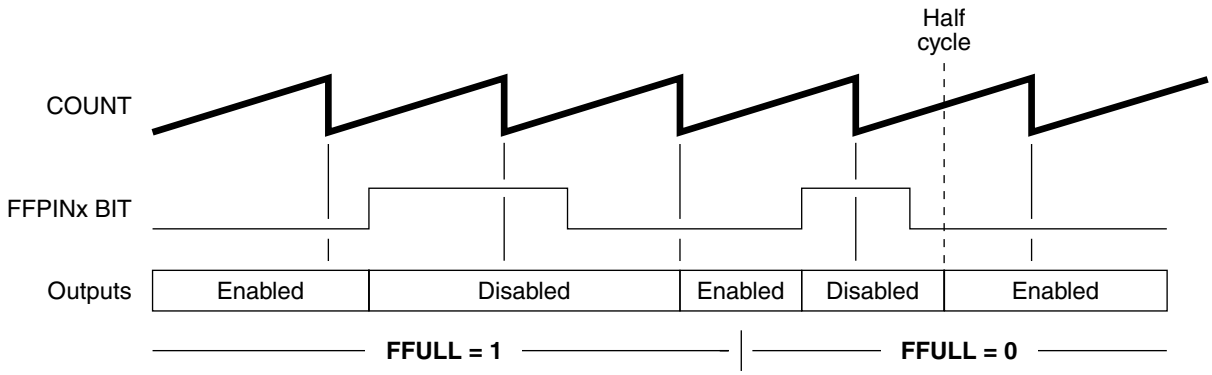


Figure 42-28. Automatic Fault Clearing

42.5.2.11.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for manual clearing:

- If the fail safe mode bits, FSAFEx, are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the following figure. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.
- If the fail safe mode bits, FSAFEx, are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag
  - The filter detects a logic one on the FAULTx pin at the start of the next PWM full or half cycle boundary. See Figure 42-30. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.

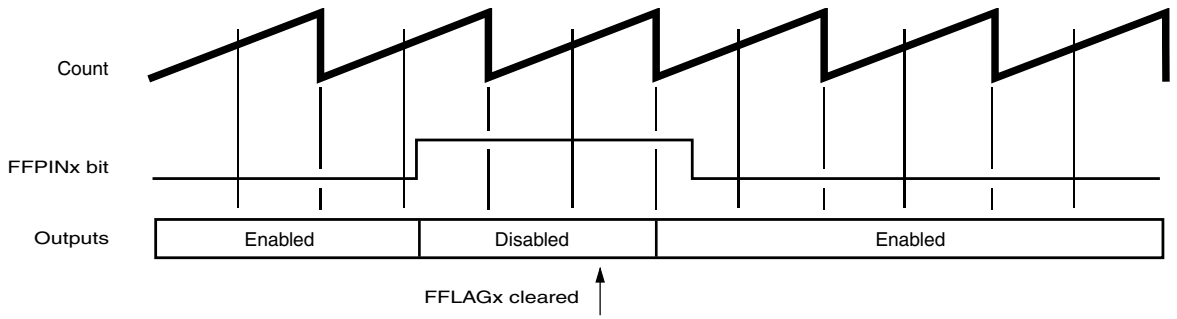


Figure 42-29. Manual Fault Clearing (FSAFE=0)

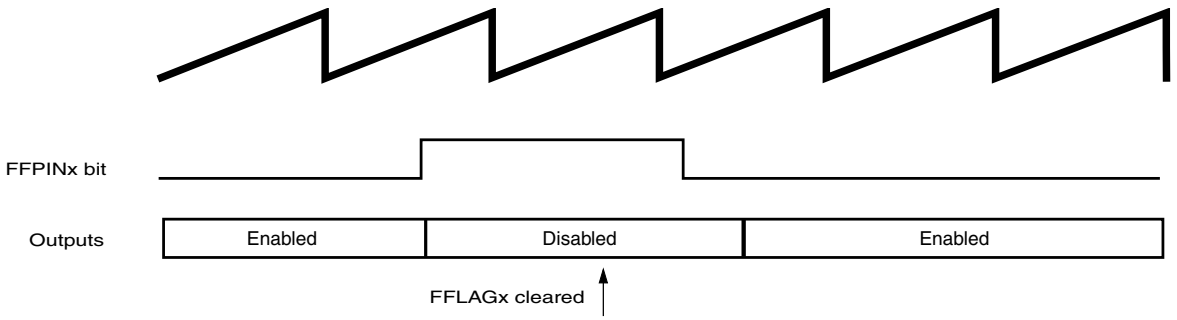


Figure 42-30. Manual Fault Clearing (FSAFE=1)

Note

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or EXTA and EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, RUN equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, RUN equals zero. Thus, fault clearing occurs at peripheral cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

42.5.2.11.4 Fault Testing

The FTEST bit is used to simulate a fault condition on each of the fault inputs.

42.5.3 PWM Generator Loading

This section describes load enabling, load frequency, the reload flag, reload errors, and initialization of the PWM generator.

42.5.3.1 Load Enable

The LDOK bit enables loading of the following PWM generator parameters:

- The prescaler divisor—from SUB*n*_CTRL1[PRSC]
- The PWM period and pulse width—from the SUB*n*_INIT and SUB*n*_VAL*x* registers

LDOK allows software to finish calculating all of these PWM parameters so they can be synchronously updated. SUB*n*_CTRL1[PRSC], SUB*n*_INIT and SUB*n*_VAL*x* are loaded by software into a set of outer buffers. When LDOK is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transferred to the inner set of registers immediately upon setting LDOK if LDMOD is set. Set LDOK by reading it when it is a logic zero and then writing a logic one to it. After loading, LDOK is automatically cleared.

42.5.3.2 Load Frequency

The LDFQ bits in the CTRL1 register select an integral loading frequency of one to 16 PWM reload opportunities. The LDFQ bits take effect at every PWM reload opportunity, regardless the state of the LDOK bit. The HALF and FULL bits in the CTRL1 register control reload timing. If FULL is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If HALF is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both HALF and FULL are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

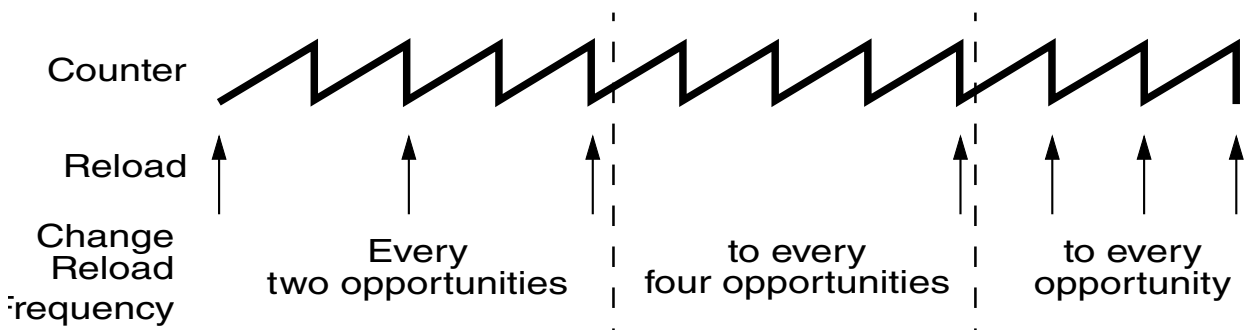


Figure 42-31. Full Cycle Reload Frequency Change

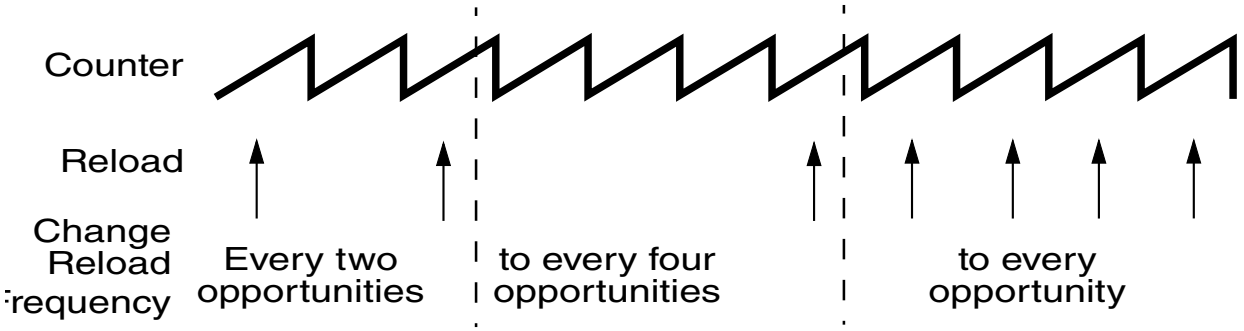


Figure 42-32. Half Cycle Reload Frequency Change

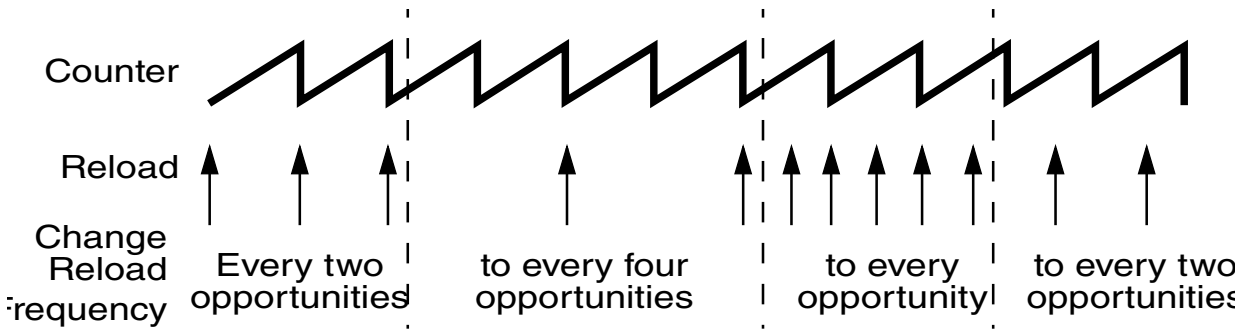


Figure 42-33. Full and Half Cycle Reload Frequency Change

42.5.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (RF) in the FlexPWM_SUBn_STS register is set. Setting RF happens even if an actual reload is prevented by the LDOK bit. If the PWM reload interrupt enable bit, RIE is set, the RF flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When RIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

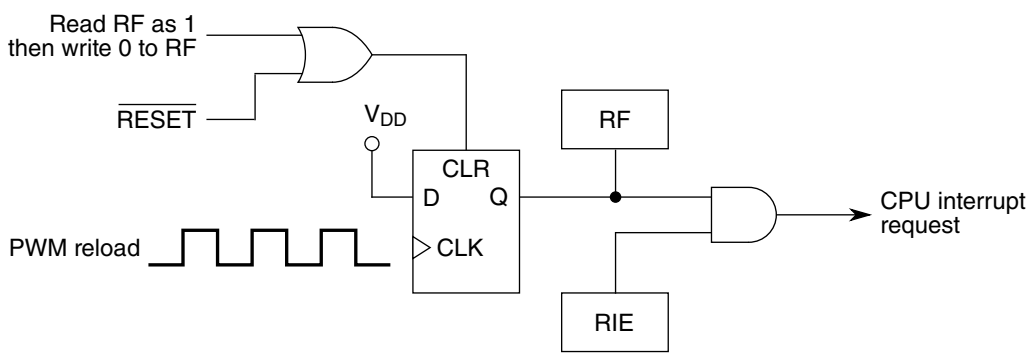


Figure 42-34. PWMF Reload Interrupt Request

42.5.3.4 Reload Errors

Whenever either SUB*n*_VAL*x* or SUB*n*_CTRL1[PSRC] is updated, the RUF flag is set to indicate that the data is not coherent. RUF will be cleared by a successful reload which consists of the reload signal while LDOK is set. If RUF is set and LDOK is clear when the reload signal occurs, a reload error has taken place and the REF bit is set. If RUF is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

42.5.3.5 Initialization

Initialize all registers and set the LDOK bit before setting the RUN bit.

Note

Even if LDOK is not set, setting RUN also sets the RF flag. To prevent a CPU interrupt request, clear the RIE bit before setting RUN.

The PWM generator uses the last values loaded if RUN is cleared and then set while LDOK equals zero.

When the RUN bit is cleared:

- The RF flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

NOTE

For any FAULT inputs not being used by the FlexPWM, the corresponding SUB*n*_DISMAP bits should be cleared.

42.6 Resets

A reset will force all registers to their reset states and tri-state the PWM outputs.

42.7 Interrupts

Each of the submodules within the FlexPWM can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 42-4. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_comp0	CMPF_0	CMPIE_0	Submodule 0 compare interrupt	Compare event has occurred
ipi_int_capt0	CFX1_0, CFX0_0	CFX1IE_0, CFX0IE_0	Submodule 0 input capture interrupt	Input capture event has occurred
ipi_int_reload0	RF_0	RIE_0	Submodule 0 reload interrupt	Reload event has occurred
ipi_int_comp1	CMPF_1	CMPIE_1	Submodule 1 compare interrupt	Compare event has occurred
ipi_int_capt1	CFX1_1, CFX0_1	CFX1IE_1, CFX0IE_1	Submodule 1 input capture interrupt	Input capture event has occurred
ipi_int_reload1	RF_1	RIE_1	Submodule 1 reload interrupt	Reload event has occurred
ipi_int_comp2	CMPF_2	CMPIE_2	Submodule 2 compare interrupt	Compare event has occurred
ipi_int_capt2	CFX1_2, CFX0_2	CFX1IE_2, CFX0IE_2	Submodule 2 input capture interrupt	Input capture event has occurred
ipi_int_reload2	RF_2	RIE_2	Submodule 2 reload interrupt	Reload event has occurred
ipi_int_comp3	CMPF_3	CMPIE_3	Submodule 3 compare interrupt	Compare event has occurred
ipi_int_capt3	CFX1_3, CFX0_3	CFX1IE_3, CFX0IE_3	Submodule 3 input capture interrupt	Input capture event has occurred
ipi_int_reload3	RF_3	RIE_3	Submodule 3 reload interrupt	Reload event has occurred
ipi_int_rerr	REF_0	REIE_0	Submodule 0 reload error interrupt	Reload error has occurred
	REF_1	REIE_1	Submodule 1 reload error interrupt	
	REF_2	REIE_2	Submodule 2 reload error interrupt	
	REF_3	REIE_3	Submodule 3 reload error interrupt	
ipi_int_fault	FFLAG	FIE	Fault input interrupt	Fault condition has been detected

42.8 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered SUB*n*_VAL*x* registers.

Table 42-5. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	CX0DE_0	Capture FIFO X0 read request	CVAL0 contains a value to be read

Table continues on the next page...

Table 42-5. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
	CX1DE_0	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_0	Capture FIFO read request source select	Selects source of read DMA request
Submodule 0 write request	VALDE_0	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 1 read request	CX0DE_1	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_1	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_1	Capture FIFO read request source select	Selects source of read DMA request
Submodule 1 write request	VALDE_1	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 2 read request	CX0DE_2	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_2	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_2	Capture FIFO read request source select	Selects source of read DMA request
Submodule 2 write request	VALDE_2	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 3 read request	CX0DE_3	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_3	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_3	Capture FIFO read request source select	Selects source of read DMA request
Submodule 3 write request	VALDE_3	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated

Chapter 43
Analog-to-Digital Converter (ADC)

43.1 Chip-specific SAR-ADC information

43.1.1 SAR-ADC Overview

There are two 12-bit Successive Approximation Analog to Digital Converter (ADC) modules located on this chip. Both ADC_0 and ADC_1 support 16 analog watchdogs. The critical and non critical faults of ADC_0 and ADC_1 are mapped to FCCU. Refer Fault inputs for details.

43.1.2 eTimer interface

Table 43-1. Interconnection between ETIMER and ADC

Input Module	Input Signal	Driving Module	Driver Signal
ADC_0	INJECT	eTimer_1	T5
ADC_1	INJECT	eTimer_1	T5

43.1.3 ADC pin muxing

The following table provides ADC pin multiplexing details.

Channel Number	ADC_0	ADC_1	Channel Type
Channel 0	PB[7] (GPIO_23)	PB[13] (GPIO_29)	External
Channel 1	PB[8] (GPIO_24)	PB[14] (GPIO_30)	External
Channel 2	PC[1] (GPIO_33)	PB[15] (GPIO_31)	External
Channel 3	PC[2] (GPIO_34)	PC[0] (GPIO_32)	External
Channel 4	PE[6] (GPIO_70)		External

Table continues on the next page...

Channel 5	PE[2] (GPIO_66)		External
Channel 6	PE[7] (GPIO_71)		External
Channel 7	PE[4] (GPIO_68)		External
Channel 8	PE[5] (GPIO_69)		External
Channel 9			Internal
Channel 10	Bandgap Reference PMC [1]	Bandgap Reference PMC [1]	Internal
Channel 11	PB[9] (GPIO_25)	PB[9] (GPIO_25)	External
Channel 12	PB[10] GPIO_26)	PB[10] (GPIO_26)	External
Channel 13	PB[11] (GPIO_27)	PB[11] (GPIO_27)	External
Channel 14	PB[12] (GPIO_28)	PB[12] (GPIO_28)	External
Channel 15	TSENS0 [2]	TSENS1 [2]	Internal

1. For sampling bandgap reference PMC, set the CTR0[INSAMP] = 0x50 minimum given an ADC clock of 80 MHz. This gives a minimum sample time of 1 microsecond.
2. For sampling TSENSn, set the CTR1[INSAMP] = 0xFF minimum given an ADC clock of 80 MHz.

NOTE

Any reserved channel if tried to be converted would give garbage value.

43.1.4 ADC channel conversion

For ADC channel conversion:

- Both Vref VDD_HV_ADCREF1 and VDD_HV_ADCREF2 must be powered up.
- Both must be at the same level.

43.1.5 ADC_0 valid control registers and bits

Table 43-2. ADC_0 valid control registers and bits

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_MCR	R	OWREN	WLSIDE	MODE	0	Reserved	Reserved	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	CTUEN	0
	W																
	R	STCL	0	0	0	0	0	0	ADCLKSEL	ABORT_CHAIN	ABORT	ACKO	0	0	REFSEL		PWDN
	W																

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_MSR	R	CALIBRTD	0	0	0	0	0	0	NSTART	JABORT	0	0	JSTART	0	SELF_TEST_S	0	CTUSTART
	W																
	R	CHADDR							0	0	0	ACKO	0	0	ADCSTATUS		
	W																
ADC_0_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOCTU	JEOC	JECH	EOC	ECH
	W												w1c	w1c	w1c	w1c	w1c
ADC_0_CEOCFR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	Reserved	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKEOC	MSKECH
	W																
ADC_0_CIMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	Reserved	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_WTISR	R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_WTIMR	R	MSKWDG15H	MSKWDG15H	MSKWDG14H	MSKWDG14H	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	MSKWDG7H	MSKWDG7H	MSKWDG6H	MSKWDG6H	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ADC_0_DMAR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	Reserved	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
	W																
ADC_0_THRHLR0-3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_PSCR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	Reserved		Reserved		PREVAL0		PRECONV
	W																
ADC_0_PSR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	Reserved	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
	W																
ADC_0_CTR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_0_CTR1[1]	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_0_NCMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	Reserved	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W																
ADC_0_JCMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	Reserved	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W																
ADC_0_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ADC_0_CDR*n*	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	CDATA															
	W																
ADC_0_THRHLR4-5	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_CWSELR0	R	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4			
	W																
	R	WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
	W																
ADC_0_CWSELR1	R	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12			
	W																
	R	WSEL_CH11				WSEL_CH10				Reserved				WSEL_CH8			
	W																

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_CWENR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	Reserved	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
	W																
ADC_0_AWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	Reserved	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_STCR1	R	INPSAMP_C								Reserved							
	W																
	R	INPSAMP_S								Reserved							
	W																
ADC_0_STCR2	R	0	0	0	0	MSKWDSER		MSKWDTERR	0	MSKST_EOC	Reserved				MSKWDG_EOA_C	Reserved	MSKWDG_EOA_S
	W						SER										
	R	MSKERR_C	Reserved	MSKERR_S2	MSKERR_S1	MSKERR_S0	0	0	0	EN	0	0	FMA_WDSERR	FMA_WDTERR	FMA_C	Reserved	FMA_S
	W																
ADC_0_STCR3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ALG		0	0	0	MSTEP				
	W																
ADC_0_STBRR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WDT		
	W																
	R	0	0	0	0	0	0	0	0	BR							
	W																

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_STSR1	R	0	0	0	0	WDSERR	0	WDTERR	OVERWR	ST_EOC	0	0	0	0	WDG_EOA_C	0	WDG_EOA_S
	W					w1c		w1c	w1c	w1c					w1c		w1c
	R	ERR_C	0	ERR_S2	ERR_S1	ERR_S0	0	STEP_C					0	0	0	0	0
	W	w1c		w1c	w1c	w1c											
ADC_0_STSR2	R	OVFL	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_0_STSR3	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_0_STSR4	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	0											
	W																
ADC_0_STDR1	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OWERWR	0	0
	W																
	R	0	0	0	0	TCDATA											
	W																
ADC_0_STDR2	R	FDATA												VALID	OWERWR	0	0
	W																
	R	0	0	0	0	IDATA											
	W																
ADC_0_STAW0R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																

Table continues on the next page...

Table 43-2. ADC_0 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_STAW1AR	R	AWDE	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW1BR	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW2R	R	AWDE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW4R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW5R	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_CALBISTREG[2]	R	OPMODE			TSAMP	Reserved		Reserved									
	W																
	R	C_T_BUSY	Reserved	Reserved						Reserved	NR_SMPL	AVG_EN	TEST_FAIL	Reserved			TEST_EN
	W													w1c			
ADC_0_OFSGNUSR[3]	R	0	0	0	0	0	0	GAIN_USER									
	W																
	R	0	0	0	0	0	0	OFFSET_USER									
	W																

1. This register affects only ADC_0 Ch. 15 (TSENS0)
2. CALBISTREG retains value after all Long Functional Resets except the one caused by External reset event. It also retains value after Short Functional Resets.
3. OFSGNUSR retains value after all Long Functional Resets except the one caused by External reset event. It also retains value after Short Functional Resets.

43.1.6 ADC_1 valid control registers and bits

NOTE

User should not overwrite the default values

Table 43-3. ADC_1 valid control registers and bits

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_MCR	R	OWREN	WLSIDE	MODE	0	Reserved	Reserved	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	CTUEN	0
	W																
	R	STCL	0	0	0	0	0	0	ADCLKSEL	ABORT_CHAIN	ABORT	ACKO	0	0	REFSEL		PWDN
	W																
ADC_1_MSR	R	CALIBRTD	0	0	0	0	0	0	NSTART	JABORT	0	0	JSTART	0	SELF_TEST_S	0	CTUSTART
	W																
	R	CHADDR							0	0	0	ACKO	0	0	ADCSTATUS		
	W																
ADC_1_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOCTU	JEOC	JECH	EOC	ECH
	W												w1c	w1c	w1c	w1c	w1c
ADC_1_CEOCFR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	EOCCH3	EOCCH2	EOCCH1	EOCCH0
	W	w1c	w1c	w1c	w1c	w1c	w1c							w1c	w1c	w1c	w1c
ADC_1_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKEOC	MSKECH
	W																
ADC_1_CIMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CIM3	CIM2	CIM1	CIM0
	W	w1c	w1c	w1c	w1c	w1c	w1c							w1c	w1c	w1c	w1c
ADC_1_WTISR	R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_WTIMR	R	MSKWDG15H	MSKWDG15H	MSKWDG14H	MSKWDG14H	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	MSKWDG7H	MSKWDG7H	MSKWDG6H	MSKWDG6H	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ADC_1_DMAR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DMA3	DMA2	DMA1	DMA0
	W																

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_THRHLR0-3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_PSCR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	Reserved		Reserved		PREVAL0		PRECONV
	W																
ADC_1_PSR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	PRES3	PRES2	PRES1	PRES0
	W																
ADC_1_CTR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_1_CTR1[1]	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_1_NCMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CH3	CH2	CH1	CH0
	W																
ADC_1_JCMR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CH3	CH2	CH1	CH0
	W																
ADC_1_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_CDR*n*	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	CDATA															
	W																
ADC_1_THRHLR4-5	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_CWSELR0	R	Reserved				Reserved				Reserved				Reserved			
	W																
	R	WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
	W																
ADC_1_CWSELR1	R	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12			
	W																
	R	WSEL_CH11				WSEL_CH10				0				Reserved			
	W																
ADC_1_CWENR0	R	0	0	0	0	0	0	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	W																
	R	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CWEN3	CWEN2	CWEN1	CWEN0
	W																
ADC_1_AWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c							w1c	w1c	w1c	w1c
ADC_1_STCR1	R	INPSAMP_C								Reserved							
	W																
	R	INPSAMP_S								Reserved							
	W																

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_STCR2	R	0	0	0	0	MSKWDSERR	SERR	MSKWDTERR	0	MSKST_EO	Reserved				MSKWDG_EOA_C	Reserved	MSKWDG_EOA_S
	W																
	R	MSKERR_C	Reserved	MSKERR_S2	MSKERR_S1	MSKERR_S0	0	0	0	EN	0	0	FMA_WDSERR	FMA_WDTERR	FMA_C	Reserved	FMA_S
	W																
ADC_1_STCR3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ALG		0	0	0	MSTEP				
	W																
ADC_1_STBRR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WDT		
	W																
	R	0	0	0	0	0	0	0	0	BR							
	W																
ADC_1_STSR1	R	0	0	0	0	WDSERR	0	WDTERR	OVERWR	ST_EOC	0	0	0	0	WDG_EOA_C	0	WDG_EOA_S
	W					w1c		w1c	w1c	w1c					w1c		w1c
	R	ERR_C	0	ERR_S2	ERR_S1	ERR_S0	0	STEP_C					0	0	0	0	0
	W	w1c		w1c	w1c	w1c											
ADC_1_STSR2	R	OVFL	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_1_STSR3	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_1_STSR4	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	0											

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
ADC_1_STDR1	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OWERWR	0	0
	W																
	R	0	0	0	0	TCDATA											
	W																
ADC_1_STDR2	R	FDATA												VALID	OWERWR	0	0
	W																
	R	0	0	0	0	IDATA											
	W																
ADC_1_STAW0R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_STAW1AR	R	AWDE	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_STAW1BR	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_STAW2R	R	AWDE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_STAW4R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_STAW5R	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																

Table continues on the next page...

Table 43-3. ADC_1 valid control registers and bits (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_CALBISTREG[2]	R	OPMODE			TSAMP		Reserved			Reserved							
	W																
	R	C_T_BUSY	Reserved		Reserved					Reserved	NR_SMPL	AVG_EN	TEST_FAIL		Reserved		TEST_EN
	W													w1c			
ADC_1_OFSGNUSR[3]	R	0	0	0	0	0	0	GAIN_USER									
	W																
	R	0	0	0	0	0	0	OFFSET_USER									
	W																

1. This register affects only ADC_1 Ch. 15 (TSENS1)
2. CALBISTREG retains value after all Long Functional Resets except the one caused by External reset event. It also retains value after Short Functional Resets.
3. OFSGNUSR retains value after all Long Functional Resets except the one caused by External reset event. It also retains value after Short Functional Resets.

43.2 Introduction

The analog-to-digital converter (ADC) uses Successive Approximation Register (SAR) architecture and provides accurate and fast conversions for a wide range of applications.

43.3 Overview

43.3.1 Parts of the ADC

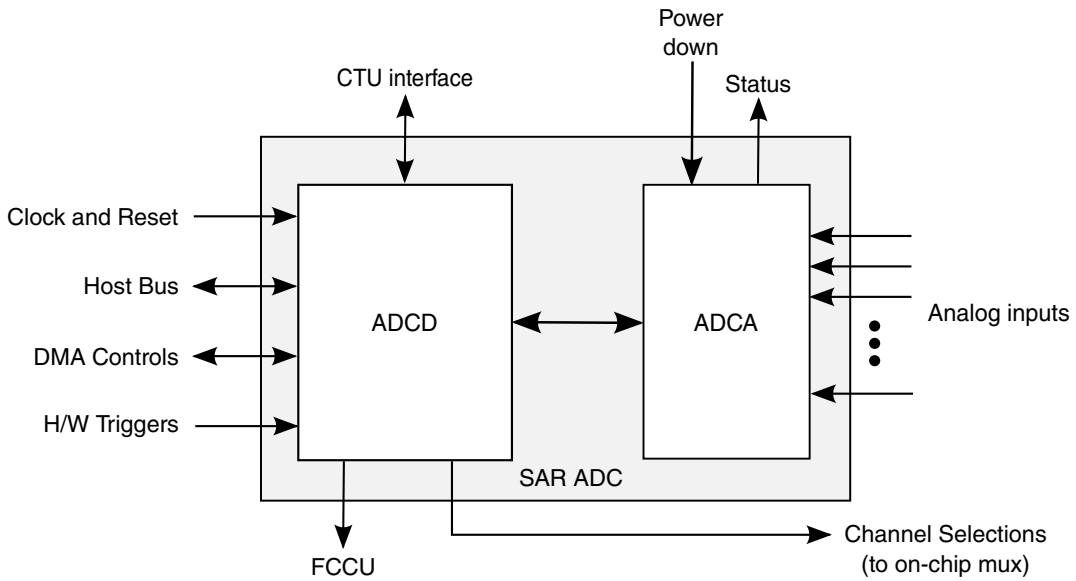


Figure 43-1. ADC high-level block diagram

The ADC is comprised of:

- An ADC digital interface (ADCD). The ADCD holds the configuration, control, and status registers accessible to software via the host bus. The converted value is stored in a register and then transferred through the DMA or host access.
- An ADC analog hard macrocell (ADCA). The analog channel inputs are fed to the inputs of the ADCA. Each channel is sampled for a specific duration and compared with an analog voltage generated with digital code via a digital-to-analog converter (DAC) in the ADCA. The comparison result is given to the ADCD to generate the converted digital value.

43.3.2 ADC clock signals

The ADC uses the following clock signals:

- Bus clock—bus clock at ADCD. Used to access the registers and the operating clock of inter-module interfaces.
- AD_clk—derived from *bus clock*, this signal serves as the operating clock for the ADCA and the SAR controller inside the ADCD.

The relation between the two clocks is determined by the setting of the ADC_MCR[ADCLKSEL] field.

- If ADC_MCR[ADCLKSEL] is 1, the clock signals are the same
- If ADC_MCR[ADCLKSEL] is 0, AD_clk is half of the bus clock.

NOTE

1. Regardless of the setting of the ADC_MCR[ADCLKSEL] field, the maximum clock speed for the ADC clock (AD_clk) is 80 MHz.
2. Depending on your chip's configuration and timing needs, MCR[ADCLKSEL] may always need to be 1.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 1, the ADC clock (AD_clk) is 80 MHz.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 0, the ADC clock (AD_clk) is 40 MHz.

See ADCLKSEL for details on the ADC_MCR register fields.

43.3.3 About ADC channels

43.3.3.1 Channel configuration

ADC channels can be configured in the following ways:

- The sampling duration is controlled independently, through programming the Conversion Timing Register (CTR).
- Separate mask registers can be programmed to configure the channels to be converted (each bit in these registers represents one channel).
- Temperature Sensor (TSENS) and other special channels (band gap, supply etc.) are available on specific channels. (See the chip-specific configuration for availability.)
- Analog watchdogs allow continuous hardware monitoring of the converted value from input channels. (See the chip-specific configuration for availability.)
- Capacitive self test
- Presampling

43.3.3.2 Initiating conversions

Conversions can be initiated by either software or hardware.

The ADCD has an on-chip Cross-Triggering Unit (CTU) interface that can also initiate conversions. The CTU interface supports CTU Control mode.

An example of an application for which CTU triggering is suitable is a current control loop for lamps or LEDs. Using an on-chip timer module as trigger source, a CTU can initiate periodic conversions.

43.3.3.3 Interrupt/DMA support

The ADCD provides interrupt/DMA support for each type of channel for various end-of-channel conversion conditions. Data can be transferred via DMA.

43.3.3.4 Self-test

The ADCD can be configured to periodically check the health of the ADCA through various self-tests and communicate any critical/non-critical faults to an on-chip Fault Collection and Control Unit (FCCU) if available (See the chip-specific section for availability). The severity (critical/noncritical) of the different tests is programmable.

43.4 Features

The following general features are implemented in the ADC. (See the chip-specific section for availability of features, as each instance of an ADC within a chip may support different features.)

- Resolution: 12-bit
- Maximum speed: 1 Mega samples/second at 80 MHz
- 16 channels
- Sampling time register
  - *precision* channels:
    - 0 - 15 (ADC0)
    - 0 - 15 (ADC1)

NOTE

The Temperature Sensor uses the value in the CTR1 register

- Modes of operation:
  - Normal

    - Injected
    - CTU
- Normal mode supports One-Shot/Scan (continuous)
- Injected mode supports One-Shot
- Presampling
- Power-Down mode
- Two abort features that allow you to abort either a single channel in a chain or the full chain in normal and injected mode
- Each channel has a dedicated register for conversion results and mode of operation (normal, injected, or CTU)
- Analog watchdog support:
    - Up to 16 analog watchdogs
    - Configurable for different channels
- Auto clock-off feature
- Programmable clock prescaler (bus clock divided by 2)
- CTU Control mode
- DMA support for each channel
- Interrupts for the following conditions:
    - End of conversion for a single channel for both normal and injected conversions
    - End of conversion for a chain for both normal and injected conversions
    - End of CTU conversion
    - Watchdog thresholds crossover
- Software-initiated calibration
- Self-testing feature

43.5 Memory Map/Register Definition

The following tables describe the registers and bit meanings for this module.

NOTE

There will be an Access Error for Reserved and out-of-range addresses. Special exceptions are the Reserved spaces at 039Ch and 03A4h: Do not write or change any value in these locations, or the ADC might malfunction.

NOTE

See the chip-specific configuration information because not all registers are available in all ADC instances.

Control logic registers:

- Main Configuration Register (ADC_MCR)
- Main Status register (ADC_MSR)
- Interrupt Status Register (ADC_ISR)
- Channel Pending register 0 (ADC_CEOCFR0)
- Interrupt Mask Register (ADC_IMR)
- Channel Interrupt Mask Register 0 (ADC_CIMR0)
- Watchdog Threshold Interrupt Status Register (ADC_WTISR)
- Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)
- Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

DMA registers:
- DMA Enable register (ADC_DMAE)
- DMA Channel Select Register 0 (ADC_DMAR0)

Threshold registers:

These registers are used to store the user programmable lower and upper thresholds 12-bit values. These registers include:
- Threshold Register (ADC_THRHLR*n*)

Presampling registers:
- Presampling Control Register (ADC_PSCR)
- Presampling register 0 (ADC_PSR0)

Conversion timing registers:

- Conversion Timing Register 0 (ADC_CTR0)
- Conversion Timing Register 1 (ADC_CTR1)

Mask register: These registers are used to program which input channels must be converted during Normal and Injected conversion. These registers include:
- Normal Conversion Mask Register 0 (ADC_NCMR0)
- Injected Conversion Mask Register 0 (ADC_JCMR0)

Delay registers:
- Power Down Exit Delay Register (ADC_PDEDR)

Data registers:

The conversion results for the internal channels are loaded into data registers. Each data register also gives information regarding the corresponding result. These registers start with CDR0:
- Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Self test registers:
- Self Test Configuration Register 1 (ADC_STCR1)

- Self Test Configuration Register 2 (ADC_STCR2)
- Self Test Configuration Register 3 (ADC_STCR3)
- Self Test Status Register 1 (ADC_STSR1)
- Self Test Status Register 2 (ADC_STSR2)
- Self Test Status Register 3 (ADC_STSR3)
- Self Test Status Register 4 (ADC_STSR4)
- Self Test Baud Rate Register (ADC_STBRR)
- Self Test Analog Watchdog Register 0 (ADC_STAW0R)
- Self Test Analog Watchdog Register 4 (ADC_STAW4R)
- Self Test Analog Watchdog Register 1A (ADC_STAW1AR)
- Self Test Analog Watchdog Register 1B (ADC_STAW1BR)
- Self Test Analog Watchdog Register 2 (ADC_STAW2R)
- Self Test Analog Watchdog Register 5 (ADC_STAW5R)
- Self Test Data Register 1 (ADC_STDR1)
- Self Test Data Register 2 (ADC_STDR2)

Other registers include:

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
- Offset and Gain User Register (ADC_OFSGNUSR)

ADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Main Configuration Register (ADC_MCR)	32	R/W	0000_0101h	43.5.1/1671
4	Main Status register (ADC_MSR)	32	R	0000_0001h	43.5.2/1674
10	Interrupt Status Register (ADC_ISR)	32	R/W	0000_0000h	43.5.3/1676
14	Channel Pending register 0 (ADC_CEOCFR0)	32	R/W	0000_0000h	43.5.4/1677
20	Interrupt Mask Register (ADC_IMR)	32	R/W	0000_0000h	43.5.5/1679
24	Channel Interrupt Mask Register 0 (ADC_CIMR0)	32	R/W	0000_0000h	43.5.6/1679
30	Watchdog Threshold Interrupt Status Register (ADC_WTISR)	32	R/W	0000_0000h	43.5.7/1682
34	Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	32	R/W	0000_0000h	43.5.8/1685
40	DMA Enable register (ADC_DMAE)	32	R/W	0000_0000h	43.5.9/1688
44	DMA Channel Select Register 0 (ADC_DMAR0)	32	R/W	0000_0000h	43.5.10/ 1689
60	Threshold Register (ADC_THRHLR0)	32	R/W	0FFF_0000h	43.5.11/ 1690
64	Threshold Register (ADC_THRHLR1)	32	R/W	0FFF_0000h	43.5.11/ 1690
68	Threshold Register (ADC_THRHLR2)	32	R/W	0FFF_0000h	43.5.11/ 1690
6C	Threshold Register (ADC_THRHLR3)	32	R/W	0FFF_0000h	43.5.11/ 1690

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	Presampling Control Register (ADC_PSCR)	32	R/W	0000_0000h	43.5.12/ 1691
84	Presampling register 0 (ADC_PSR0)	32	R/W	0000_0000h	43.5.13/ 1692
94	Conversion Timing Register 0 (ADC_CTR0)	32	R/W	0000_0016h	43.5.14/ 1693
98	Conversion Timing Register 1 (ADC_CTR1)	32	R/W	0000_0016h	43.5.15/ 1694
A4	Normal Conversion Mask Register 0 (ADC_NCMR0)	32	R/W	0000_0000h	43.5.16/ 1695
B4	Injected Conversion Mask Register 0 (ADC_JCMR0)	32	R/W	0000_0000h	43.5.17/ 1696
C8	Power Down Exit Delay Register (ADC_PDEDR)	32	R/W	0000_0000h	43.5.18/ 1698
100	Channel Data Register n (Precision Channels) (ADC_CDR0)	32	R	0000_0000h	43.5.19/ 1699
104	Channel Data Register n (Precision Channels) (ADC_CDR1)	32	R	0000_0000h	43.5.19/ 1699
108	Channel Data Register n (Precision Channels) (ADC_CDR2)	32	R	0000_0000h	43.5.19/ 1699
10C	Channel Data Register n (Precision Channels) (ADC_CDR3)	32	R	0000_0000h	43.5.19/ 1699
110	Channel Data Register n (Precision Channels) (ADC_CDR4)	32	R	0000_0000h	43.5.19/ 1699
114	Channel Data Register n (Precision Channels) (ADC_CDR5)	32	R	0000_0000h	43.5.19/ 1699
118	Channel Data Register n (Precision Channels) (ADC_CDR6)	32	R	0000_0000h	43.5.19/ 1699
11C	Channel Data Register n (Precision Channels) (ADC_CDR7)	32	R	0000_0000h	43.5.19/ 1699
120	Channel Data Register n (Precision Channels) (ADC_CDR8)	32	R	0000_0000h	43.5.19/ 1699
124	Channel Data Register n (Precision Channels) (ADC_CDR9)	32	R	0000_0000h	43.5.19/ 1699
128	Channel Data Register n (Precision Channels) (ADC_CDR10)	32	R	0000_0000h	43.5.19/ 1699
12C	Channel Data Register n (Precision Channels) (ADC_CDR11)	32	R	0000_0000h	43.5.19/ 1699
130	Channel Data Register n (Precision Channels) (ADC_CDR12)	32	R	0000_0000h	43.5.19/ 1699
134	Channel Data Register n (Precision Channels) (ADC_CDR13)	32	R	0000_0000h	43.5.19/ 1699
138	Channel Data Register n (Precision Channels) (ADC_CDR14)	32	R	0000_0000h	43.5.19/ 1699

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13C	Channel Data Register n (Precision Channels) (ADC_CDR15)	32	R	0000_0000h	43.5.19/ 1699
280	Threshold Register (ADC_THRHLR4)	32	R/W	0FFF_0000h	43.5.20/ 1700
284	Threshold Register (ADC_THRHLR5)	32	R/W	0FFF_0000h	43.5.20/ 1700
288	Threshold Register (ADC_THRHLR6)	32	R/W	0FFF_0000h	43.5.20/ 1700
28C	Threshold Register (ADC_THRHLR7)	32	R/W	0FFF_0000h	43.5.20/ 1700
290	Threshold Register (ADC_THRHLR8)	32	R/W	0FFF_0000h	43.5.20/ 1700
294	Threshold Register (ADC_THRHLR9)	32	R/W	0FFF_0000h	43.5.20/ 1700
298	Threshold Register (ADC_THRHLR10)	32	R/W	0FFF_0000h	43.5.20/ 1700
29C	Threshold Register (ADC_THRHLR11)	32	R/W	0FFF_0000h	43.5.20/ 1700
2A0	Threshold Register (ADC_THRHLR12)	32	R/W	0FFF_0000h	43.5.20/ 1700
2A4	Threshold Register (ADC_THRHLR13)	32	R/W	0FFF_0000h	43.5.20/ 1700
2A8	Threshold Register (ADC_THRHLR14)	32	R/W	0FFF_0000h	43.5.20/ 1700
2AC	Threshold Register (ADC_THRHLR15)	32	R/W	0FFF_0000h	43.5.20/ 1700
2B0	Channel Watchdog Select Register 0 (ADC_CWSELR0)	32	R/W	0000_0000h	43.5.21/ 1700
2B4	Channel Watchdog Select Register 1 (ADC_CWSELR1)	32	R/W	0000_0000h	43.5.22/ 1701
2E0	Channel Watchdog Enable Register 0 (ADC_CWENR0)	32	R/W	0000_0000h	43.5.23/ 1702
2F0	Analog Watchdog Out of Range Register 0 (ADC_AWORR0)	32	R/W	0000_0000h	43.5.24/ 1704
340	Self Test Configuration Register 1 (ADC_STCR1)	32	R/W	1818_2507h	43.5.25/ 1706
344	Self Test Configuration Register 2 (ADC_STCR2)	32	R/W	0000_0005h	43.5.26/ 1707
348	Self Test Configuration Register 3 (ADC_STCR3)	32	R/W	0000_0300h	43.5.27/ 1709
34C	Self Test Baud Rate Register (ADC_STBRR)	32	R/W	0005_0000h	43.5.28/ 1710
350	Self Test Status Register 1 (ADC_STSR1)	32	R/W	0000_0000h	43.5.29/ 1712

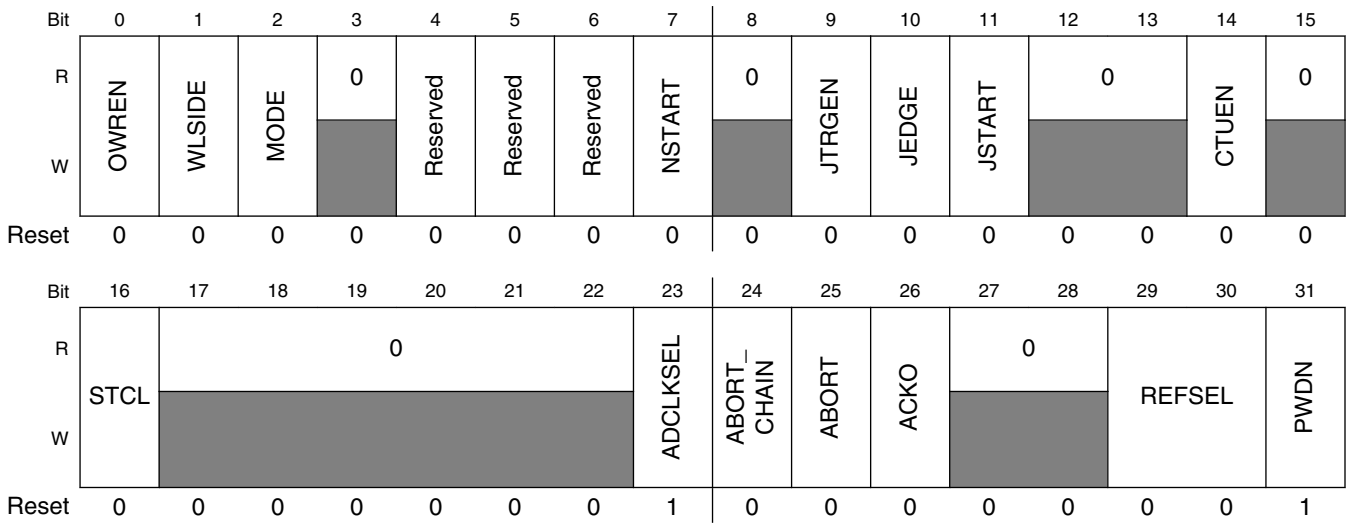
Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
354	Self Test Status Register 2 (ADC_STSR2)	32	R	0000_0000h	43.5.30/ 1715
358	Self Test Status Register 3 (ADC_STSR3)	32	R	0000_0000h	43.5.31/ 1716
35C	Self Test Status Register 4 (ADC_STSR4)	32	R	0000_0000h	43.5.32/ 1716
370	Self Test Data Register 1 (ADC_STDR1)	32	R	0000_0000h	43.5.33/ 1717
374	Self Test Data Register 2 (ADC_STDR2)	32	R	0000_0000h	43.5.34/ 1719
380	Self Test Analog Watchdog Register 0 (ADC_STAW0R)	32	R/W	0727_04C5h	43.5.35/ 1720
384	Self Test Analog Watchdog Register 1A (ADC_STAW1AR)	32	R/W	0003_0001h	43.5.36/ 1722
388	Self Test Analog Watchdog Register 1B (ADC_STAW1BR)	32	R/W	03E8_0ED0h	43.5.37/ 1723
38C	Self Test Analog Watchdog Register 2 (ADC_STAW2R)	32	R/W	0000_0FF9h	43.5.38/ 1724
394	Self Test Analog Watchdog Register 4 (ADC_STAW4R)	32	R/W	0010_0FF0h	43.5.39/ 1725
398	Self Test Analog Watchdog Register 5 (ADC_STAW5R)	32	R/W	0010_0FF0h	43.5.40/ 1726
3A0	Calibration, BIST Control and status Register (ADC_CALBISTREG)	32	R/W	C737_06F6h	43.5.41/ 1727
3A8	Offset and Gain User Register (ADC_OFSGNUSR)	32	R/W	0000_0000h	43.5.42/ 1730

43.5.1 Main Configuration Register (ADC_MCR)

Address: 0h base + 0h offset = 0h



ADC_MCR field descriptions

Field	Description
0 OWREN	Overwrite enable. 0 Older valid conversion data is not overwritten by newer conversion data. 1 Newer conversion result is always overwritten, irrespective of the validity of older conversion data.
1 WLSIDE	Write Left/Right aligned. 0 The conversion data is written right aligned from (32-resolution) to 31). 1 Data is left aligned (from 16 to (16 + resolution - 1).
2 MODE	One_Shot/Scan. 0 One_Shot Mode: configure the Normal conversion of one chain. 1 Scan Mode: configure continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. Writing a value to this field has no effect.
5 Reserved	This field is reserved. Writing a value to this field has no effect.
6 Reserved	This field is reserved. **NOTE:** User must not overwrite default value of this field.
7 NSTART	Start of Normal conversion. Setting this field starts a normal conversion. In One shot mode: This field is cleared as soon as conversion is started. Setting this field during any ongoing conversion keeps this bit high until the requested conversion is started.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	NOTE: This bit cannot be set (1) when PWDN = 1. This bit is always cleared when PWDN = 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 JTRGEN	Injection external trigger enable. When set, enables the external trigger to start a injected conversion; if the external trigger feature is needed for injected conversion then this field must be set.
10 JEDGE	Injection trigger edge selection. Edge selection for external trigger; if JTRGEN is one this bit selects the falling (JEDGE = zero) or rising (JEDGE = one) edge for the external trigger.
11 JSTART	Injection start. Setting this field will start the configured injected conversion chain. Resetting this field has no effect, as the injected chain conversion cannot be interrupted. Setting this field during ongoing Injected conversion keeps this field high value until the requested conversion is started.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTUEN	Crosstrigger Unit Enable. Enables the crosstriggering unit (CTU). 0 The crosstriggering unit is disabled and the CTU-triggered injected conversion cannot take place. 1 The crosstriggering unit is enabled and the CTUtriggered injected conversion can take place.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 STCL	Self Testing Configuration Lock. 0 No lock 1 The self-testing configuration is locked i.e. STCR1, STCR2, STCR3, STBRR, STAW0R, STAW1AR, STAW1BR, STAW2R, STAW4R, STAW5R are write-protected. It can be used only in CPU and SCAN mode. The lock bit is cleared only by a peripheral reset.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ADCLKSEL	Analog Clock frequency Selector. This field can be written in Power Down mode only. Please check device specific additional restriction on programming of this field. 0 ADC clock frequency is half of bus clock 1 ADC clock frequency is equal to bus clock
24 ABORT_CHAIN	Abort Chain. If this bit is set the ongoing Chain Conversion is aborted. This bit gets reset by hardware as soon as a new conversion is requested. **NOTE:** Setting this bit in idle state (no normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing CTU conversion this bit also cannot be programmed.
25 ABORT	Abort Conversion.

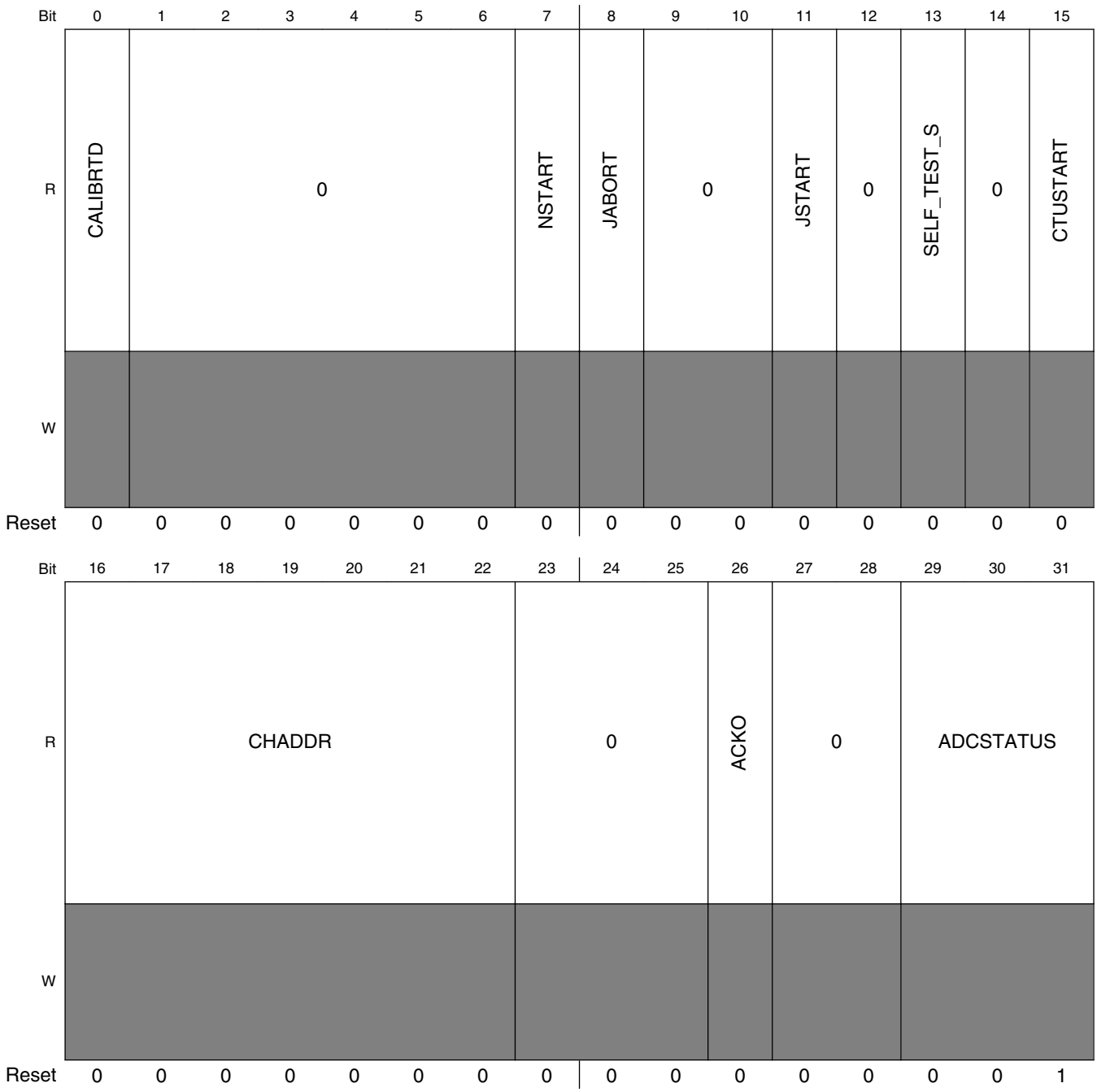
Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	If this bit is set, the ongoing channel conversion is aborted and the next channel conversion is invoked. This bit gets reset by hardware as soon as the new conversion is invoked. This bit cannot be written while self test conversion is ongoing. **NOTE:** Setting this bit in idle state (No Normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing CTU conversion this bit cannot be programmed
26 ACKO	Auto clock off enable. If set, enables the Auto clock off feature.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 REFSEL	Reference voltage selection for ADC analog part. 00 Select VREFH as reference voltage 01 Reserved 10 Reserved 11 Reserved
31 PWDN	Power-down enable. When this bit is set, the analog module is requested to enter Power-Down mode. When the ADC's status is PWDN, resetting this bit will start the ADC's transition to IDLE mode. In CTU control mode MCR[CTUEN] must be reset before entering into power-down mode.

43.5.2 Main Status register (ADC_MSR)

Address: 0h base + 4h offset = 4h



ADC_MSR field descriptions

Field	Description
0 CALIBRTD	This bit indicates the ADC calibration status. This bit gets updated after running the High Accuracy Calibration mode.

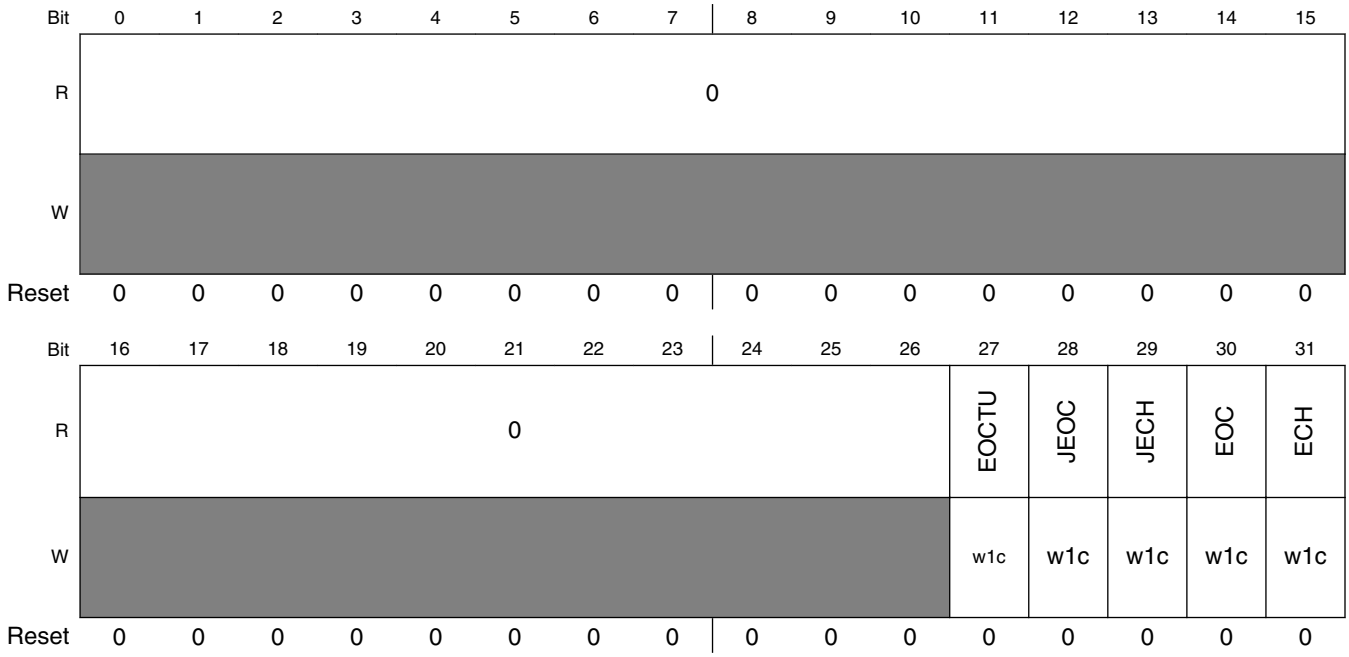
Table continues on the next page...

ADC_MSR field descriptions (continued)

Field	Description
	0 Uncalibrated or calibration unsuccessful 1 Calibration was run and it was successful
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NSTART	This status bit is used to signal that a Normal conversion is ongoing.
8 JABORT	This status bit is used to signal that an Injected conversion has been aborted by CTU. That bit is reset when a new injected conversion starts.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 JSTART	This status bit is used to signal that an Injected conversion is ongoing.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SELF_TEST_S	This status bit signals that self test conversion is ongoing.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CTUSTART	This status bit is used to signal that a CTU conversion is ongoing. This bit is set when a CTU trigger pulse is received and the CTU conversion starts.When Control Mode is enabled this bit is reset when the CTU is disabled (CTUEN set to 0).
16–22 CHADDR	Channel under measure address. Status bits are used to signal which channel is under measure. MSR[CHADDR] is only valid when the SARADC is in the sample phase of the conversion (MSR[ADCSTATUS] = 100b).
23–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ACKO	Auto clock off enable. This status bit is used to signal if the Auto clock off feature is on.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 ADCSTATUS	Status of the ADC. 000 Idle 001 Power Down 100 Sample 110 Conversion 010 Wait state (waiting to start conversion [external trigger]) 011 Busy in Calibration

43.5.3 Interrupt Status Register (ADC_ISR)

Address: 0h base + 10h offset = 10h



ADC_ISR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 EOCTU	End of CTU conversion. This bit is set for a digital end of conversion on a the CTU Channel; active when set.
28 JEOC	End of injected channel conversion. This bit is set for a digital end of conversion on an injected channel; active when set.
29 JECH	End of injected chain conversion. This bit is set for a digital end of chain conversion on an injected channel; active when set. **NOTE:** If there is no channel selected in the ADC_JCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[JECH] is set and a JECH interrupt is immediately issued (if enabled). Similarly, if no channel is selected in the ADC_NCMR*n* register and a normal conversion is triggered, ADC_ISR[ECH] is set and an ECH interrupt is immediately issued, if enabled.
30 EOC	End of channel conversion. This bit is set for a digital end of conversion on a Normal Channel; active when set.
31 ECH	End of chain conversion. This bit is set for a digital end of chain conversion on a Normal Channel; active when set. **NOTE:** If there is no channel selected in the ADC_NCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

Table continues on the next page...

ADC_ISR field descriptions (continued)

Field	Description
	Similarly, if no channel is selected in the ADC_JCMR*n* register and an injected conversion is triggered, ADC_ISR[JECH] is set and a JECH interrupt is immediately issued, if enabled.

43.5.4 Channel Pending register 0 (ADC_CEOCFR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	EOCCH9	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EOCCH15	EOC Channel 15. When set, the measure of channel 15 is completed.

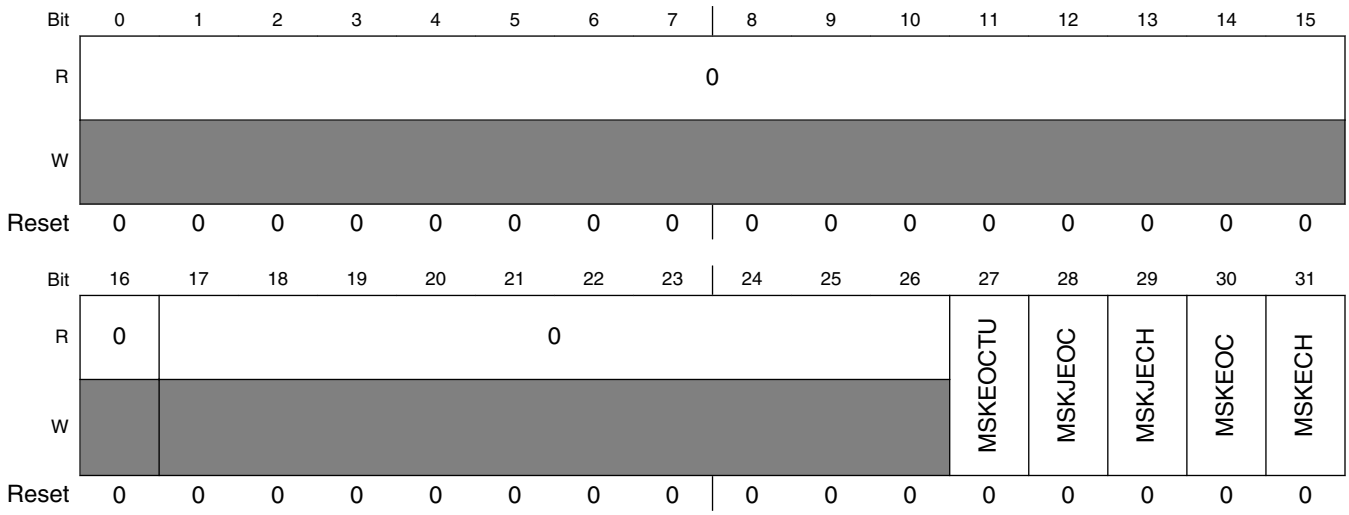
Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
17 EOCCH14	EOC Channel 14. When set, the measure of channel 14 is completed.
18 EOCCH13	EOC Channel 13. When set, the measure of channel 13 is completed.
19 EOCCH12	EOC Channel 12. When set, the measure of channel 12 is completed.
20 EOCCH11	EOC Channel 11. When set, the measure of channel 11 is completed.
21 EOCCH10	EOC Channel 10. When set, the measure of channel 10 is completed.
22 EOCCH9	EOC Channel 9. When set, the measure of channel 9 is completed.
23 EOCCH8	EOC Channel 8. When set, the measure of channel 8 is completed.
24 EOCCH7	EOC Channel 7. When set, the measure of channel 7 is completed.
25 EOCCH6	EOC Channel 6. When set, the measure of channel 6 is completed.
26 EOCCH5	EOC Channel 5. When set, the measure of channel 5 is completed.
27 EOCCH4	EOC Channel 4. When set, the measure of channel 4 is completed.
28 EOCCH3	EOC Channel 3. When set, the measure of channel 3 is completed.
29 EOCCH2	EOC Channel 2. When set, the measure of channel 2 is completed.
30 EOCCH1	EOC Channel 1. When set, the measure of channel 1 is completed.
31 EOCCH0	EOC Channel 0. When set, the measure of channel 0 is completed.

43.5.5 Interrupt Mask Register (ADC_IMR)

Address: 0h base + 20h offset = 20h



ADC_IMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MSKEOCTU	Mask bit for EOCTU. When set, the interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC. When set, the interrupt is enabled.
29 MSKJECH	Mask bit for JECH. When set, the interrupt is enabled.
30 MSKEOC	Mask bit for EOC. When set, the interrupt is enabled.
31 MSKECH	Mask bit for ECH. When set, the interrupt is enabled.

43.5.6 Channel Interrupt Mask Register 0 (ADC_CIMR0)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 0 through 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	CIM9	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CIM15	Channel 15 interrupt enable. When this field is set, the interrupt for channel 15 is enabled.
17 CIM14	Channel 14 interrupt enable. When this field is set, the interrupt for channel 14 is enabled.
18 CIM13	Channel 13 interrupt enable. When this field is set, the interrupt for channel 13 is enabled.
19 CIM12	Channel 12 interrupt enable. When this field is set, the interrupt for channel 12 is enabled.
20 CIM11	Channel 11 interrupt enable. When this field is set, the interrupt for channel 11 is enabled.
21 CIM10	Channel 10 interrupt enable. When this field is set, the interrupt for channel 10 is enabled.
22 CIM9	Channel 9 interrupt enable. When this field is set, the interrupt for channel 9 is enabled.
23 CIM8	Channel 8 interrupt enable. When this field is set, the interrupt for channel 8 is enabled.
24 CIM7	Channel 7 interrupt enable. When this field is set, the interrupt for channel 7 is enabled.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
25 CIM6	Channel 6 interrupt enable. When this field is set, the interrupt for channel 6 is enabled.
26 CIM5	Channel 5 interrupt enable. When this field is set, the interrupt for channel 5 is enabled.
27 CIM4	Channel 4 interrupt enable. When this field is set, the interrupt for channel 4 is enabled.
28 CIM3	Channel 3 interrupt enable. When this field is set, the interrupt for channel 3 is enabled.
29 CIM2	Channel 2 interrupt enable. When this field is set, the interrupt for channel 2 is enabled.
30 CIM1	Channel 1 interrupt enable. When this field is set, the interrupt for channel 1 is enabled.
31 CIM0	Channel 0 interrupt enable. When this field is set, the interrupt for channel 0 is enabled.

43.5.7 Watchdog Threshold Interrupt Status Register (ADC_WTISR)

The fields in this register, when set, indicate either the upper or lower threshold value has been crossed for a specific watchdog.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTISR field descriptions

Field	Description
0 WDG15H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 15 (WD15). **NOTE:** Write a '1' to this field to clear it.
1 WDG15L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 15 (WD15). **NOTE:** Write a '1' to this field to clear it.
2 WDG14H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 14 (WD14). **NOTE:** Write a '1' to this field to clear it.
3 WDG14L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 14 (WD14). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
4 WDG13H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 13 (WD13). **NOTE:** Write a '1' to this field to clear it.
5 WDG13L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 13 (WD13). **NOTE:** Write a '1' to this field to clear it.
6 WDG12H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 12 (WD12). **NOTE:** Write a '1' to this field to clear it.
7 WDG12L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 12 (WD12). **NOTE:** Write a '1' to this field to clear it.
8 WDG11H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 11 (WD11). **NOTE:** Write a '1' to this field to clear it.
9 WDG11L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 11 (WD11). **NOTE:** Write a '1' to this field to clear it.
10 WDG10H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 10 (WD10). **NOTE:** Write a '1' to this field to clear it.
11 WDG10L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 10 (WD10). **NOTE:** Write a '1' to this field to clear it.
12 WDG9H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 9 (WD9). **NOTE:** Write a '1' to this field to clear it.
13 WDG9L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 9 (WD9). **NOTE:** Write a '1' to this field to clear it.
14 WDG8H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 8 (WD8). **NOTE:** Write a '1' to this field to clear it.
15 WDG8L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 8 (WD8). **NOTE:** Write a '1' to this field to clear it.
16 WDG7H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 7 (WD7). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
17 WDG7L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 7 (WD7). **NOTE:** Write a '1' to this field to clear it.
18 WDG6H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 6 (WD6). **NOTE:** Write a '1' to this field to clear it.
19 WDG6L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 6 (WD6). **NOTE:** Write a '1' to this field to clear it.
20 WDG5H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
21 WDG5L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
22 WDG4H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
23 WDG4L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
24 WDG3H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
25 WDG3L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
26 WDG2H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
27 WDG2L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
28 WDG1H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
29 WDG1L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
30 WDG0H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.
31 WDG0L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.

43.5.8 Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTIMR field descriptions

Field	Description
0 MSKWDG15H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 15 (WD15) being higher than the programmed higher threshold. When set, the interrupt is enabled.
1 MSKWDG15L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 15 (WD15) being lower than the programmed lower threshold. When set, the interrupt is enabled.
2 MSKWDG14H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 14 (WD14) being higher than the programmed higher threshold. When set, the interrupt is enabled.
3 MSKWDG14L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 14 (WD14) being lower than the programmed lower threshold. When set, the interrupt is enabled.
4 MSKWDG13H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 13 (WD13) being higher than the programmed higher threshold. When set, the interrupt is enabled.

Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
5 MSKWDG13L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 13 (WD13) being lower than the programmed lower threshold. When set, the interrupt is enabled.
6 MSKWDG12H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 12 (WD12) being higher than the programmed higher threshold. When set, the interrupt is enabled.
7 MSKWDG12L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 12 (WD12) being lower than the programmed lower threshold. When set, the interrupt is enabled.
8 MSKWDG11H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 11 (WD11) being higher than the programmed higher threshold. When set, the interrupt is enabled.
9 MSKWDG11L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 11 (WD11) being lower than the programmed lower threshold. When set, the interrupt is enabled.
10 MSKWDG10H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 10 (WD10) being higher than the programmed higher threshold. When set, the interrupt is enabled.
11 MSKWDG10L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 10 (WD10) being lower than the programmed lower threshold. When set, the interrupt is enabled.
12 MSKWDG9H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 9 (WD9) being higher than the programmed higher threshold. When set, the interrupt is enabled.
13 MSKWDG9L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 9 (WD9) being lower than the programmed lower threshold. When set, the interrupt is enabled.
14 MSKWDG8H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 8 (WD8) being higher than the programmed higher threshold. When set, the interrupt is enabled.
15 MSKWDG8L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 8 (WD8) being lower than the programmed lower threshold. When set, the interrupt is enabled.
16 MSKWDG7H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 7 (WD7) being higher than the programmed higher threshold. When set, the interrupt is enabled.
17 MSKWDG7L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 7 (WD7) being lower than the programmed lower threshold. When set, the interrupt is enabled.
18 MSKWDG6H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 6 (WD6) being higher than the programmed higher threshold. When set, the interrupt is enabled.
19 MSKWDG6L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 6 (WD6) being lower than the programmed lower threshold. When set, the interrupt is enabled.
20 MSKWDG5H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being higher than the programmed higher threshold. When set, the interrupt is enabled.

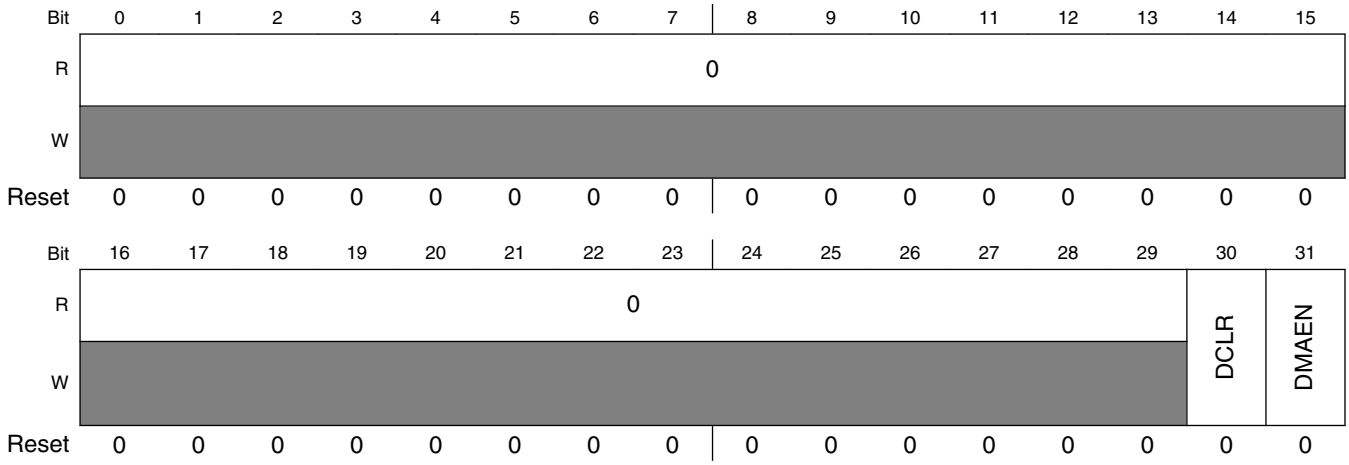
Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
21 MSKWDG5L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being lower than the programmed lower threshold. When set, the interrupt is enabled.
22 MSKWDG4H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being higher than the programmed higher threshold. When set, the interrupt is enabled.
23 MSKWDG4L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being lower than the programmed lower threshold. When set, the interrupt is enabled.
24 MSKWDG3H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being higher than the programmed higher threshold. When set, the interrupt is enabled.
25 MSKWDG3L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being lower than the programmed lower threshold. When set, the interrupt is enabled.
26 MSKWDG2H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being higher than the programmed higher threshold. When set, the interrupt is enabled.
27 MSKWDG2L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being lower than the programmed lower threshold. When set, the interrupt is enabled.
28 MSKWDG1H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being higher than the programmed higher threshold. When set, the interrupt is enabled.
29 MSKWDG1L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being lower than the programmed lower threshold. When set, the interrupt is enabled.
30 MSKWDG0H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being higher than the programmed higher threshold. When set, the interrupt is enabled.
31 MSKWDG0L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being lower than the programmed lower threshold. When set, the interrupt is enabled.

43.5.9 DMA Enable register (ADC_DMAE)

Address: 0h base + 40h offset = 40h



ADC_DMAE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DCLR	DMA Clear sequence enable. 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers
31 DMAEN	DMA global enable. When this bit is set, the DMA feature is enabled.

43.5.10 DMA Channel Select Register 0 (ADC_DMAR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	DMA9	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 DMA15	DMA enable for channel 15. When set, transferring data in DMA mode is enabled for channel 15.
17 DMA14	DMA enable for channel 14. When set, transferring data in DMA mode is enabled for channel 14.
18 DMA13	DMA enable for channel 13. When set, transferring data in DMA mode is enabled for channel 13.
19 DMA12	DMA enable for channel 12. When set, transferring data in DMA mode is enabled for channel 12.
20 DMA11	DMA enable for channel 11. When set, transferring data in DMA mode is enabled for channel 11.
21 DMA10	DMA enable for channel 10. When set, transferring data in DMA mode is enabled for channel 10.
22 DMA9	DMA enable for channel 9.

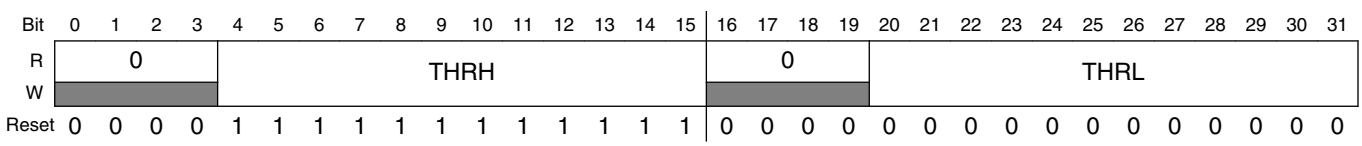
Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 9.
23 DMA8	DMA enable for channel 8. When set, transferring data in DMA mode is enabled for channel 8.
24 DMA7	DMA enable for channel 7. When set, transferring data in DMA mode is enabled for channel 7.
25 DMA6	DMA enable for channel 6. When set, transferring data in DMA mode is enabled for channel 6.
26 DMA5	DMA enable for channel 5. When set, transferring data in DMA mode is enabled for channel 5.
27 DMA4	DMA enable for channel 4. When set, transferring data in DMA mode is enabled for channel 4.
28 DMA3	DMA enable for channel 3. When set, transferring data in DMA mode is enabled for channel 3.
29 DMA2	DMA enable for channel 2. When set, transferring data in DMA mode is enabled for channel 2.
30 DMA1	DMA enable for channel 1. When set, transferring data in DMA mode is enabled for channel 1.
31 DMA0	DMA enable for channel 0. When set, transferring data in DMA mode is enabled for channel 0.

43.5.11 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 60h offset + (4d × i), where i=0d to 3d



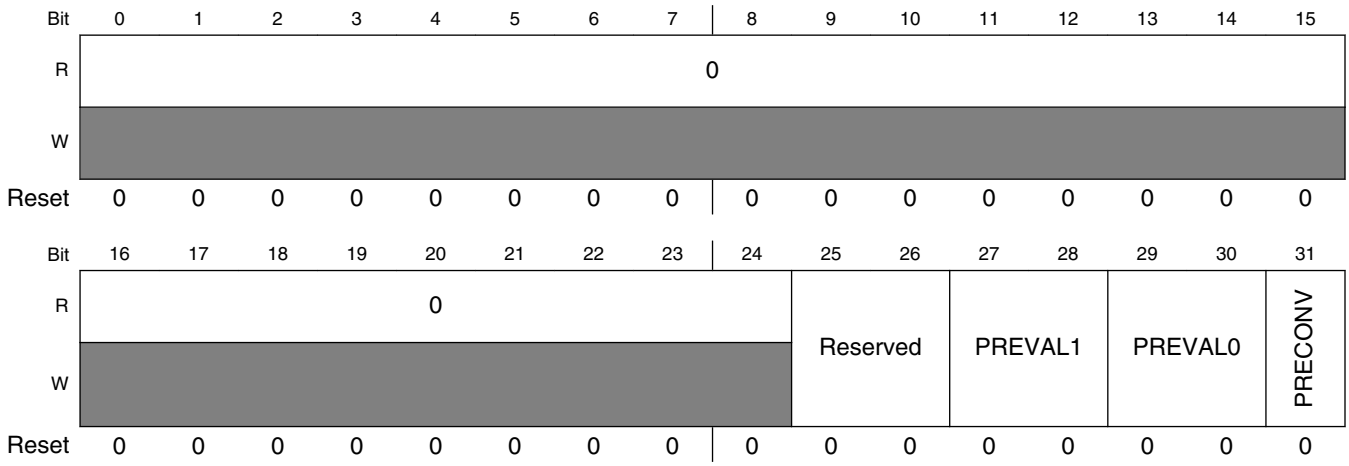
ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for watchdog n.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for watchdog n.

43.5.12 Presampling Control Register (ADC_PSCR)

See the chip configuration details for the availability of this register.

Address: 0h base + 80h offset = 80h



ADC_PSCR field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–26 Reserved	This field is reserved.
27–28 PREVAL1	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available internal voltages using internal extended channels (channel 32 to 63). This is used for temperature sensor also. Refer to Presampling channel enable.
29–30 PREVAL0	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available four internal voltages for Internal precision channels (Channel 0 to 31). Refer to Presampling channel enable.
31 PRECONV	Convert Presampled value If bit PRECONV is set, presampling is followed by the conversion. Sampling will be bypassed and conversion of presampled data will be done. This bit has no effect on conversion of a channel when presampling on that channel is disabled (for example, PSR*x*[PRES*y*] = 0 , where *x* = register index and *y* = channel number).

43.5.13 Presampling register 0 (ADC_PSR0)

See the chip configuration details for the availability of this register.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 PRES15	Presampling enable for channel 15. When set, presampling is enabled for channel 15.
17 PRES14	Presampling enable for channel 14. When set, presampling is enabled for channel 14.
18 PRES13	Presampling enable for channel 13. When set, presampling is enabled for channel 13.
19 PRES12	Presampling enable for channel 12. When set, presampling is enabled for channel 12.
20 PRES11	Presampling enable for channel 11. When set, presampling is enabled for channel 11.
21 PRES10	Presampling enable for channel 10. When set, presampling is enabled for channel 10.
22 PRES9	Presampling enable for channel 9. When set, presampling is enabled for channel 9.
23 PRES8	Presampling enable for channel 8. When set, presampling is enabled for channel 8.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
24 PRES7	Presampling enable for channel 7. When set, presampling is enabled for channel 7.
25 PRES6	Presampling enable for channel 6. When set, presampling is enabled for channel 6.
26 PRES5	Presampling enable for channel 5. When set, presampling is enabled for channel 5.
27 PRES4	Presampling enable for channel 4. When set, presampling is enabled for channel 4.
28 PRES3	Presampling enable for channel 3. When set, presampling is enabled for channel 3.
29 PRES2	Presampling enable for channel 2. When set, presampling is enabled for channel 2.
30 PRES1	Presampling enable for channel 1. When set, presampling is enabled for channel 1.
31 PRES0	Presampling enable for channel 0. When set, presampling is enabled for channel 0.

43.5.14 Conversion Timing Register 0 (ADC_CTR0)

Conversion timing register. CTR0 is associated with channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								INPSAMP							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0

ADC_CTR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration.

Table continues on the next page...

ADC_CTR0 field descriptions (continued)

Field	Description
	This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles. **NOTE:** The Bandgap voltage sampling time is controlled by CTR0[INPSAMP]. CTR0[INPSAMP] is also used for the external channel sample time. When the user software needs to sample the Bandgap voltage, it must change the CTR0[INPSAMP] value to allow for a longer time while the Bandgap is being sampled, then change its value back to the normal value for sampling external channels.

43.5.15 Conversion Timing Register 1 (ADC_CTR1)

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								INPSAMP							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0

ADC_CTR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

43.5.16 Normal Conversion Mask Register 0 (ADC_NCMR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH15	Normal sampling enable for channel 15. When set, normal sampling is enabled for channel 15.
17 CH14	Normal sampling enable for channel 14. When set, normal sampling is enabled for channel 14.
18 CH13	Normal sampling enable for channel 13. When set, normal sampling is enabled for channel 13.
19 CH12	Normal sampling enable for channel 12. When set, normal sampling is enabled for channel 12.
20 CH11	Normal sampling enable for channel 11. When set, normal sampling is enabled for channel 11.
21 CH10	Normal sampling enable for channel 10. When set, normal sampling is enabled for channel 10.
22 CH9	Normal sampling enable for channel 9.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 9.
23 CH8	Normal sampling enable for channel 8. When set, normal sampling is enabled for channel 8.
24 CH7	Normal sampling enable for channel 7. When set, normal sampling is enabled for channel 7.
25 CH6	Normal sampling enable for channel 6. When set, normal sampling is enabled for channel 6.
26 CH5	Normal sampling enable for channel 5. When set, normal sampling is enabled for channel 5.
27 CH4	Normal sampling enable for channel 4. When set, normal sampling is enabled for channel 4.
28 CH3	Normal sampling enable for channel 3. When set, normal sampling is enabled for channel 3.
29 CH2	Normal sampling enable for channel 2. When set, normal sampling is enabled for channel 2.
30 CH1	Normal sampling enable for channel 1. When set, normal sampling is enabled for channel 1.
31 CH0	Normal sampling enable for channel 0. When set, normal sampling is enabled for channel 0.

43.5.17 Injected Conversion Mask Register 0 (ADC_JCMR0)

Enable bits of Injected Sampling for channel 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH15	Injected sampling enable for channel 15. When set, injected sampling is enabled for channel 15.
17 CH14	Injected sampling enable for channel 14. When set, injected sampling is enabled for channel 14.
18 CH13	Injected sampling enable for channel 13. When set, injected sampling is enabled for channel 13.
19 CH12	Injected sampling enable for channel 12. When set, injected sampling is enabled for channel 12.
20 CH11	Injected sampling enable for channel 11. When set, injected sampling is enabled for channel 11.
21 CH10	Injected sampling enable for channel 10. When set, injected sampling is enabled for channel 10.
22 CH9	Injected sampling enable for channel 9. When set, injected sampling is enabled for channel 9.
23 CH8	Injected sampling enable for channel 8. When set, injected sampling is enabled for channel 8.
24 CH7	Injected sampling enable for channel 7. When set, injected sampling is enabled for channel 7.
25 CH6	Injected sampling enable for channel 6. When set, injected sampling is enabled for channel 6.
26 CH5	Injected sampling enable for channel 5. When set, injected sampling is enabled for channel 5.
27 CH4	Injected sampling enable for channel 4. When set, injected sampling is enabled for channel 4.
28 CH3	Injected sampling enable for channel 3. When set, injected sampling is enabled for channel 3.
29 CH2	Injected sampling enable for channel 2. When set, injected sampling is enabled for channel 2.

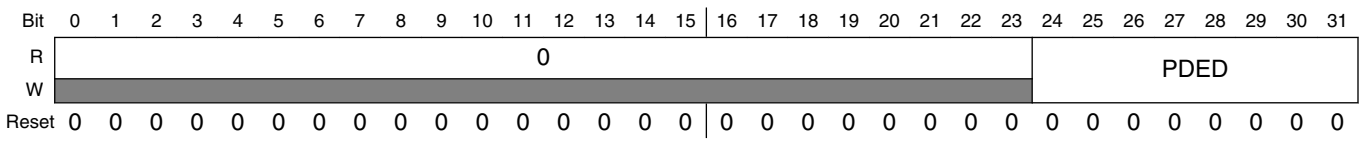
Table continues on the next page...

ADC_JCMR0 field descriptions (continued)

Field	Description
30 CH1	Injected sampling enable for channel 1. When set, injected sampling is enabled for channel 1.
31 CH0	Injected sampling enable for channel 0. When set, injected sampling is enabled for channel 0.

43.5.18 Power Down Exit Delay Register (ADC_PDEDR)

Address: 0h base + C8h offset = C8h



ADC_PDEDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PDED	The delay between the power down bit reset and the start of conversion. The power down delay is calculated as (PDED x 1/[ADC_clock_frequency]).

43.5.19 Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 100h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

43.5.20 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 280h offset + (4d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for channel x.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x.

43.5.21 Channel Watchdog Select Register 0 (ADC_CWSELR0)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B0h offset = 2B0h

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R W	WSEL_CH7	WSEL_CH6	WSEL_CH5	WSEL_CH4	WSEL_CH3	WSEL_CH2	WSEL_CH1	WSEL_CH0
Reset	0000	0000	0000	0000	0000	0000	0000	0000

ADC_CWSELR0 field descriptions

Field	Description
0–3 WSEL_CH7	Channel Watchdog select for channel 7. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH6	Channel Watchdog select for channel 6. See WSEL_CH7.
8–11 WSEL_CH5	Channel Watchdog select for channel 5. See WSEL_CH7.
12–15 WSEL_CH4	Channel Watchdog select for channel 4. See WSEL_CH7.
16–19 WSEL_CH3	Channel Watchdog select for channel 3. See WSEL_CH7.
20–23 WSEL_CH2	Channel Watchdog select for channel 2. See WSEL_CH7.
24–27 WSEL_CH1	Channel Watchdog select for channel 1. See WSEL_CH7.
28–31 WSEL_CH0	Channel Watchdog select for channel 0. See WSEL_CH7.

43.5.22 Channel Watchdog Select Register 1 (ADC_CWSELR1)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B4h offset = 2B4h

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R W	WSEL_CH15	WSEL_CH14	WSEL_CH13	WSEL_CH12	WSEL_CH11	WSEL_CH10	WSEL_CH9	WSEL_CH8
Reset	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ADC_CWSELR1 field descriptions

Field	Description
0–3 WSEL_CH15	Channel Watchdog select for channel 15. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH14	Channel Watchdog select for channel 14. See WSEL_CH15.
8–11 WSEL_CH13	Channel Watchdog select for channel 13. See WSEL_CH15.
12–15 WSEL_CH12	Channel Watchdog select for channel 12. See WSEL_CH15.
16–19 WSEL_CH11	Channel Watchdog select for channel 11. See WSEL_CH15.
20–23 WSEL_CH10	Channel Watchdog select for channel 10. See WSEL_CH15.
24–27 WSEL_CH9	Channel Watchdog select for channel 9. See WSEL_CH15.
28–31 WSEL_CH8	Channel Watchdog select for channel 8. See WSEL_CH15.

43.5.23 Channel Watchdog Enable Register 0 (ADC_CWENR0)

Controls whether the watchdog is enabled for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	CWEN9	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CWEN15	Watchdog enable for channel 15. When set, Watchdog feature is enabled for channel 15.
17 CWEN14	Watchdog enable for channel 14. When set, Watchdog feature is enabled for channel 14.
18 CWEN13	Watchdog enable for channel 13. When set, Watchdog feature is enabled for channel 13.
19 CWEN12	Watchdog enable for channel 12. When set, Watchdog feature is enabled for channel 12.
20 CWEN11	Watchdog enable for channel 11. When set, Watchdog feature is enabled for channel 11.
21 CWEN10	Watchdog enable for channel 10. When set, Watchdog feature is enabled for channel 10.
22 CWEN9	Watchdog enable for channel 9. When set, Watchdog feature is enabled for channel 9.
23 CWEN8	Watchdog enable for channel 8. When set, Watchdog feature is enabled for channel 8.
24 CWEN7	Watchdog enable for channel 7. When set, Watchdog feature is enabled for channel 7.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
25 CWEN6	Watchdog enable for channel 6. When set, Watchdog feature is enabled for channel 6.
26 CWEN5	Watchdog enable for channel 5. When set, Watchdog feature is enabled for channel 5.
27 CWEN4	Watchdog enable for channel 4. When set, Watchdog feature is enabled for channel 4.
28 CWEN3	Watchdog enable for channel 3. When set, Watchdog feature is enabled for channel 3.
29 CWEN2	Watchdog enable for channel 2. When set, Watchdog feature is enabled for channel 2.
30 CWEN1	Watchdog enable for channel 1. When set, Watchdog feature is enabled for channel 1.
31 CWEN0	Watchdog enable for channel 0. When set, Watchdog feature is enabled for channel 0.

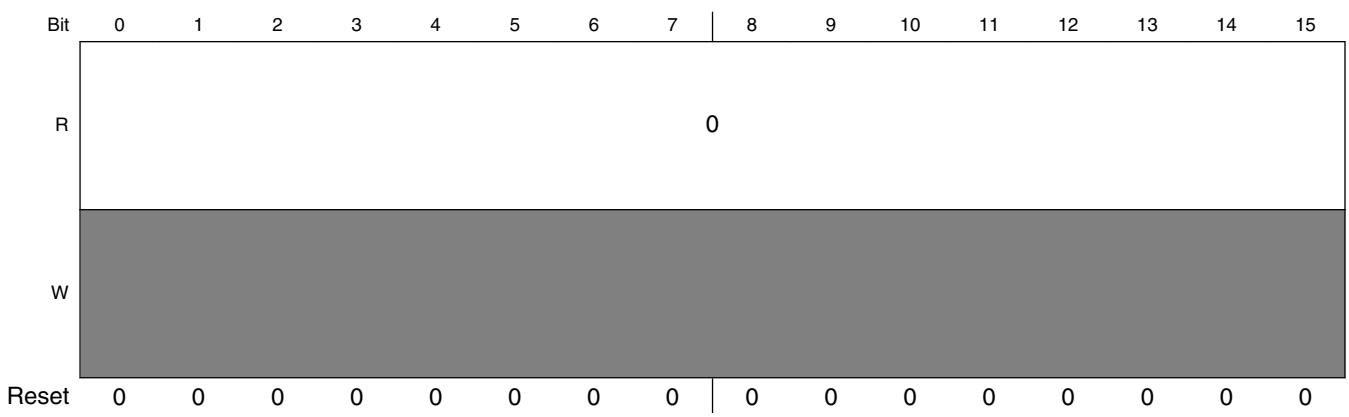
43.5.24 Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

Analog watchdog out of range register for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F0h offset = 2F0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 AWOR_CH15	Analog watchdog out of range for channel 15. When set, indicates the converted data is out of range for channel 15.
17 AWOR_CH14	Analog watchdog out of range for channel 14. When set, indicates the converted data is out of range for channel 14.
18 AWOR_CH13	Analog watchdog out of range for channel 13. When set, indicates the converted data is out of range for channel 13.
19 AWOR_CH12	Analog watchdog out of range for channel 12. When set, indicates the converted data is out of range for channel 12.
20 AWOR_CH11	Analog watchdog out of range for channel 11. When set, indicates the converted data is out of range for channel 11.
21 AWOR_CH10	Analog watchdog out of range for channel 10. When set, indicates the converted data is out of range for channel 10.
22 AWOR_CH9	Analog watchdog out of range for channel 9. When set, indicates the converted data is out of range for channel 9.
23 AWOR_CH8	Analog watchdog out of range for channel 8. When set, indicates the converted data is out of range for channel 8.
24 AWOR_CH7	Analog watchdog out of range for channel 7. When set, indicates the converted data is out of range for channel 7.
25 AWOR_CH6	Analog watchdog out of range for channel 6. When set, indicates the converted data is out of range for channel 6.
26 AWOR_CH5	Analog watchdog out of range for channel 5. When set, indicates the converted data is out of range for channel 5.
27 AWOR_CH4	Analog watchdog out of range for channel 4. When set, indicates the converted data is out of range for channel 4.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
28 AWOR_CH3	Analog watchdog out of range for channel 3. When set, indicates the converted data is out of range for channel 3.
29 AWOR_CH2	Analog watchdog out of range for channel 2. When set, indicates the converted data is out of range for channel 2.
30 AWOR_CH1	Analog watchdog out of range for channel 1. When set, indicates the converted data is out of range for channel 1.
31 AWOR_CH0	Analog watchdog out of range for channel 0. When set, indicates the converted data is out of range for channel 0.

43.5.25 Self Test Configuration Register 1 (ADC_STCR1)

Address: 0h base + 340h offset = 340h

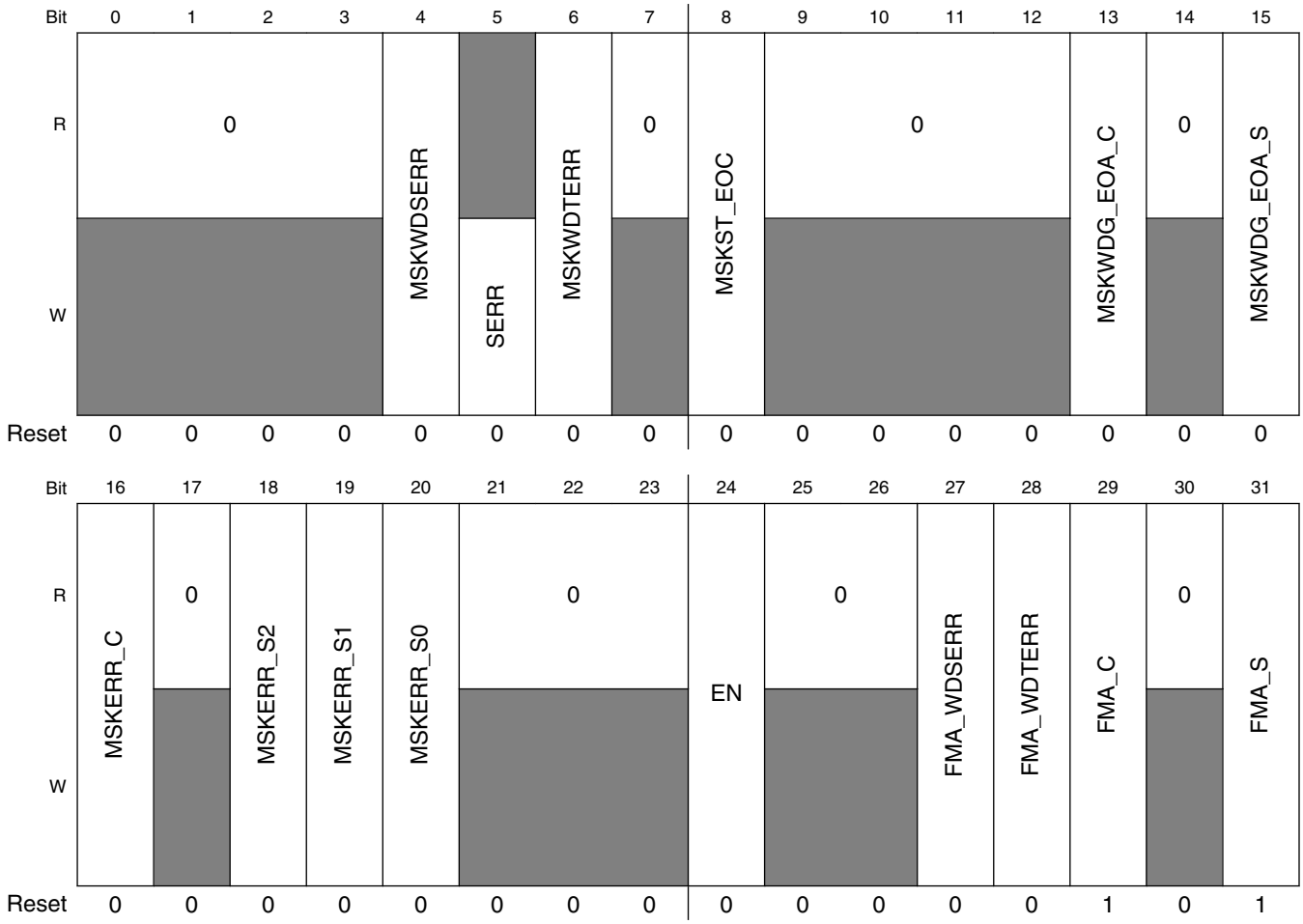
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INPSAMP_C								Reserved								INPSAMP_S								0					Reserved		
W																																
Reset	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1

ADC_STCR1 field descriptions

Field	Description
0–7 INPSAMP_C	Sampling phase duration for the test conversions related to self test algorithm C. Minimum value should be set as per default (18h). Maximum can be FFh.
8–15 Reserved	This field is reserved.
16–23 INPSAMP_S	Sampling phase duration for the test conversions related to self test algorithm S. Minimum value should be set as per default (25h). Maximum can be FFh.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

43.5.26 Self Test Configuration Register 2 (ADC_STCR2)

Address: 0h base + 344h offset = 344h



ADC_STCR2 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MSKWDSERR	Interrupt enable (STSR1[WDSERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDSERR] status bit to generate an interrupt indication.
5 SERR	Error fault injection bit (write only). Writing 1 to this bit sets the STSR1[ERRx] status bits once only. Reading always returns 0. Setting this bit does not affect any of the self test data registers.
6 MSKWDTERR	Interrupt enable (STSR[WDTERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDTERR] status bit to generate an interrupt indication.

Table continues on the next page...

ADC_STCR2 field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MSKST_EOC	Interrupt Enable bit for STSR2[ST_EOC] 0 Interrupt disabled 1 Enables the STSR1[ST_EOC] status bit to generate an interrupt indication. (IMR[MSKEOC] must also be enabled.)
9–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MSKWDG_EOA_C	Interrupt enable (WDG_EOA_C status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_C] status bit to generate an interrupt indication.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 MSKWDG_EOA_S	Interrupt enable (WDG_EOA_S status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_S] status bit to generate an interrupt indication.
16 MSKERR_C	Interrupt enable (ERR_C status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_C] status bit to generate an interrupt indication.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MSKERR_S2	Interrupt enable (ERR_S2 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S2] status bit to generate an interrupt indication.
19 MSKERR_S1	Interrupt enable (ERR_S1 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S1] status bit to generate an interrupt indication.
20 MSKERR_S0	Interrupt enable (ERR_S0 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S0] status bit to generate an interrupt indication.
21–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 EN	Self testing channel enable. The TEST conversions are enabled. Enables the TEST channel only in CPU mode. This bit should be set before starting the normal conversion and should not be changed while conversion is ongoing. This bit should be reset only after end of conversion for the last self test channel has been received. 0 Disabled 1 Enabled
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

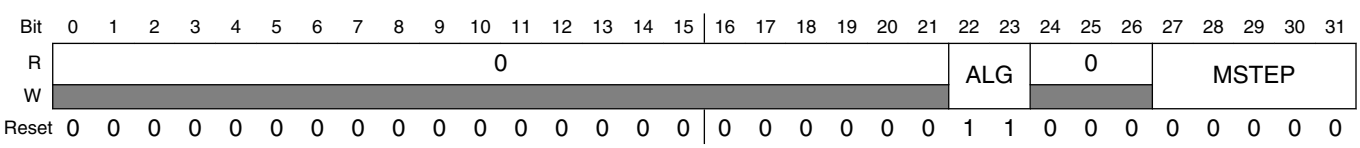
ADC_STCR2 field descriptions (continued)

Field	Description
27 FMA_WDSERR	Fault mapping for the Watchdog Sequence error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
28 FMA_WDTERR	Fault mapping for the Watchdog Timer error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
29 FMA_C	Fault mapping for self test algorithm C. By default a failure on the self test C algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FMA_S	Fault mapping for the self test algorithm BGAP. By default a failure on the Supply algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping

43.5.27 Self Test Configuration Register 3 (ADC_STCR3)

This register defines ADC self testing capabilities in CPU mode or CTU mode.

Address: 0h base + 348h offset = 348h



ADC_STCR3 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ALG	Algorithm scheduling. ONE-SHOT mode algorithm scheduling:

Table continues on the next page...

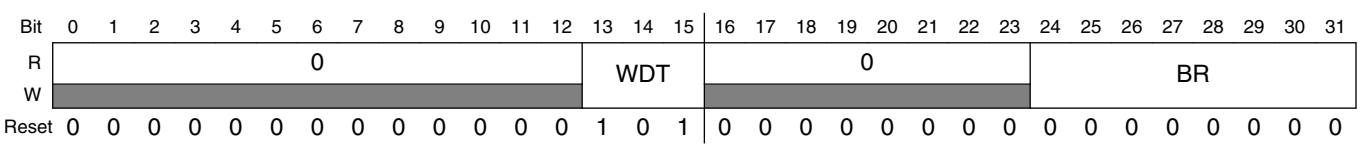
ADC_STCR3 field descriptions (continued)

Field	Description
	• 00: Algorithm S (single step=MSTEP) • 01: Reserved • 10: Algorithm C (single step=MSTEP) • 11: Algorithm S (default) For test/debug purposes. SCAN mode algorithm scheduling: • 00: Algorithm S • 01: Reserved • 10: Algorithm C • 11: Algorithm S + Algorithm C (default) The baud rate for the execution of the selected algorithm is defined by the STBRR register.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 MSTEP	One shot mode: Current step for Algorithm S or Algorithm C. • MSTEP = 0 to (NUM_SUPPLY_STEPS-1) for algorithm S • MSTEP = 0 to (NUM_C_STEPS-1) for algorithm C Scan mode: This field is not used in scan mode as always single step execution (interleaved mode) is performed. It should be programmed to zero in scan mode. **NOTE:** All unused valued are RESERVED and should not be used.

43.5.28 Self Test Baud Rate Register (ADC_STBRR)

Used to set the baud rate for the selected baud rate in SCAN mode as well as the watchdog timer value.

Address: 0h base + 34Ch offset = 34Ch



ADC_STBRR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WDT	The watchdog timer value is used to monitor that the algorithm sequence is correctly executed within the safe time period. The self testing watchdog is enabled setting the STAWxR[WDTE] control bits. Default value is 10 ms. A fixed pre-scaler runs on the ADCdig clock (80 MHz). (Prescaler value is fixed at 1024). 000 0.1 ms ((0008h * Prescaler) cycles at 80 MHz) 001 0.5 ms ((0027h * Prescaler) cycles at 80 MHz)

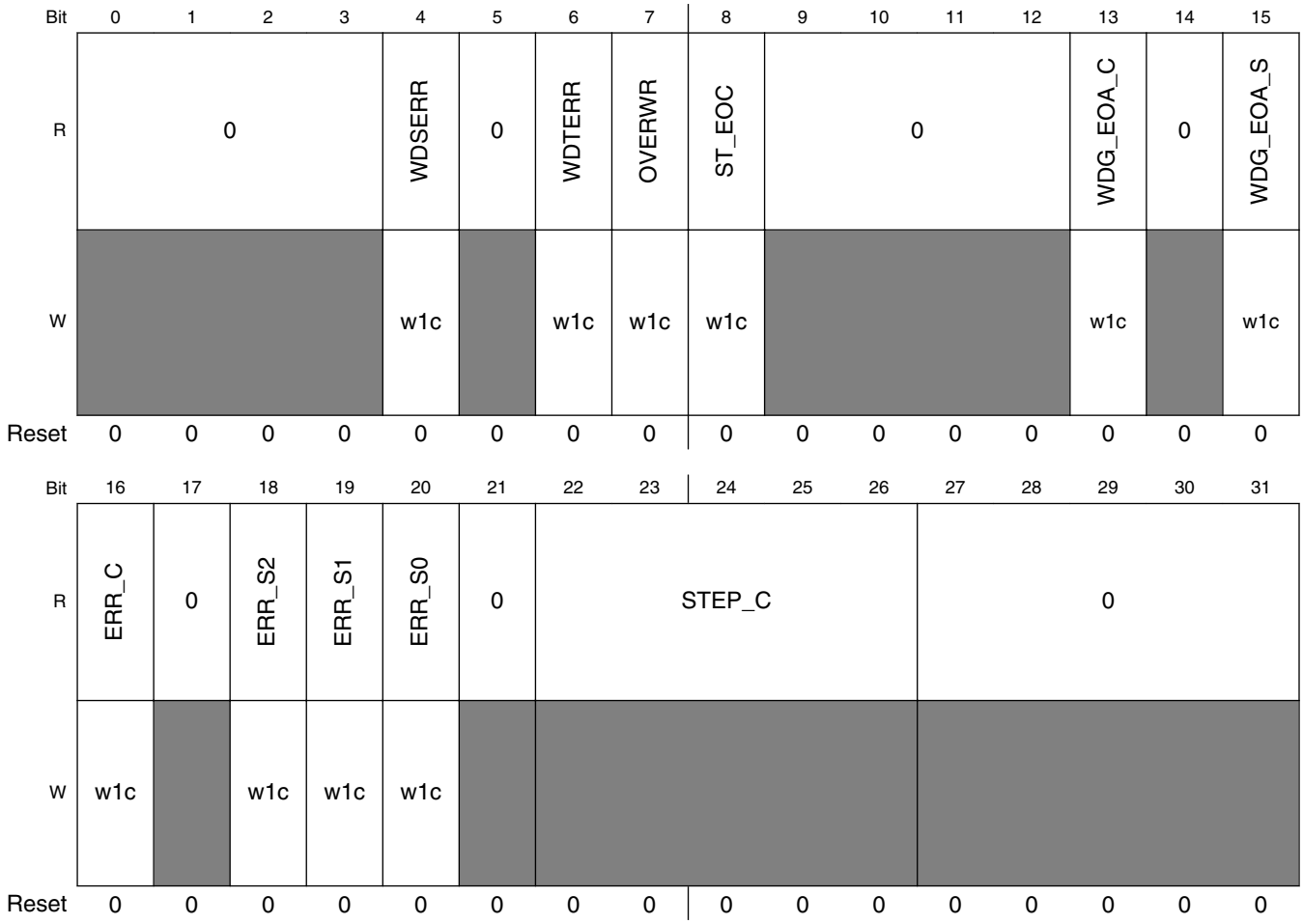
Table continues on the next page...

ADC_STBRR field descriptions (continued)

Field	Description
	010 1 ms ((004Eh * Prescaler) cycles at 80 MHz) 011 2 ms ((009Ch * Prescaler) cycles at 80 MHz) 100 5 ms ((0187h * Prescaler) cycles at 80 MHz) 101 10 ms ((030Dh * Prescaler) cycles at 80 MHz) 110 20 ms ((061Ah * Prescaler) cycles at 80 MHz) 111 50 ms ((0F42h * Prescaler) cycles at 80 MHz)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BR	Baud rate for the selected algorithm in SCAN mode (MCR[MODE] = 1). 00h Max scheduling rate (NOMINAL rate) ... FFh Min scheduling rate (NOMINAL rate scaled by 255) This field should be programmed when ST_EN is zero i.e. before enabling self test channel.

43.5.29 Self Test Status Register 1 (ADC_STSR1)

Address: 0h base + 350h offset = 350h



ADC_STSR1 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WDSERR	Watchdog sequence error of the ADC sub-system (check for algorithm step sequence). The WDSERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDSERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDSERR] mapping. **NOTE:** This bit is set only if STAWxR[WDTE] = '1'. 0 No failure 1 Failure occurred
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STSR1 field descriptions (continued)

Field	Description
6 WDTERR	Watchdog timer error of the ADC sub-system (algorithm check for completion within safe time). The WDTERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDTERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDTERR] mapping. 0 No failure 1 Failure occurred
7 OVERWR	Overwrite error. Used to notify when the STSR1[ERRx] bit is overwritten by a newer one. The OVERWR status bit is cleared writing 1. A 0 write is an invariant operation. The new error status is written or discarded according to the MCR[OWREN] bit value. To avoid OVERWR indication, the ERRx status bit must be cleared (via software).
8 ST_EOC	Self Test EOC Bit. This bit is set along with EOC bit when end_of_conversion signal is received from ADC analog for self test channel. The ST_EOC status bit is cleared by writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled by STCR2[MSKST_EOC]. (Also, IMR[MSKEOC] has to be enabled.)
9–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 WDG_EOA_C	This bit indicates that Algorithm C has been completed. This bit is generated after the last STEP of algorithm C is executed. The WDG_EOA_C status bit is cleared by writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_C] = 1). This bit is set only if STAW4R[WDTE] = 1. For CTU conversions, this bit is significant only for Burst mode of operation.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 WDG_EOA_S	This bit indicates that Algorithm S has been completed. This bit is generated after the last STEP of algorithm S is executed. The WDG_EOA_S status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_S] = 1). This bit is set only if STAW0R[WDTE] = ‘1’. For CTU conversions, this bit is significant only for Burst mode of operation.
16 ERR_C	Indicates an error on the self testing channel (algorithm C). The ERR_C status bit is cleared writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_C] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_C bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical type or Non-Critical type lines are asserted according to the STCR2[FMAx] mapping. 0 No C-algorithm ERROR 1 C-algorithm ERROR occurred
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ERR_S2	Indicates an error on the self testing channel (algorithm SUPPLY, step2). The ERR_S2 status bit is cleared by writing a 1b. A 0b write is an invariant operation. It generates an interrupt (“ipi_int_er” line) if enabled (STCR2[MSKERR_S2] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S2 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No ERROR occurred on the sampled signal 1 ERROR occurred on the sampled signal

Table continues on the next page...

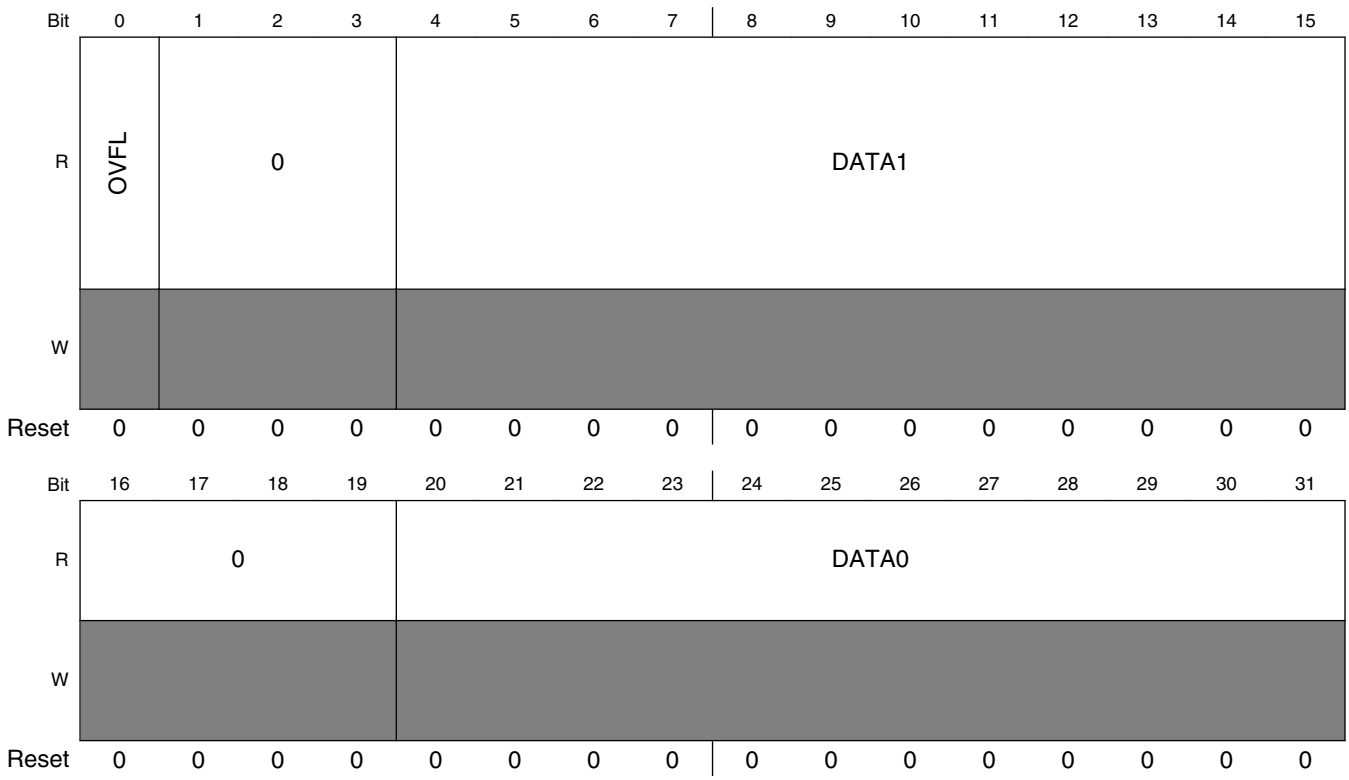
ADC_STSR1 field descriptions (continued)

Field	Description
19 ERR_S1	Indicates an error on the self testing channel (algorithm SUPPLY, step1). The ERR_S1 status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_S1] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S1 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No VDD ERROR 1 VDD ERROR occurred
20 ERR_S0	Indicates an error on the self testing channel (algorithm SUPPLY, step0). The ERR_S0 status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_S0] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S0 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical type fault lines are asserted according to the STCR2[FMAx] mapping. 0 No VREF ERROR 1 VREF ERROR occurred
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–26 STEP_C	Step of the algorithm C when an ERR_C has occurred. 0.. (NUM_C_STEPS-1) => algorithm C
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.30 Self Test Status Register 2 (ADC_STSR2)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 354h offset = 354h



ADC_STSR2 field descriptions

Field	Description
0 OVFL	Overflow Bit This bit is set when the divisor is zero. Also, the STSR1[ERR_S1] bit is set if overflow occurs.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA0	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP

43.5.31 Self Test Status Register 3 (ADC_STSR3)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 358h offset = 358h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				DATA0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S2 has occurred. Algorithm S (step2) => TEST channel data = VREF/VREF
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA0	Test channel converted data when the ERR_S0 has occurred. Algorithm S (step0) => TEST channel data = VREF/VBGAP

43.5.32 Self Test Status Register 4 (ADC_STSR4)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 35Ch offset = 35Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR4 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_C has occurred. Algorithm C => TEST channel data
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STSR4 field descriptions (continued)

Field	Description
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

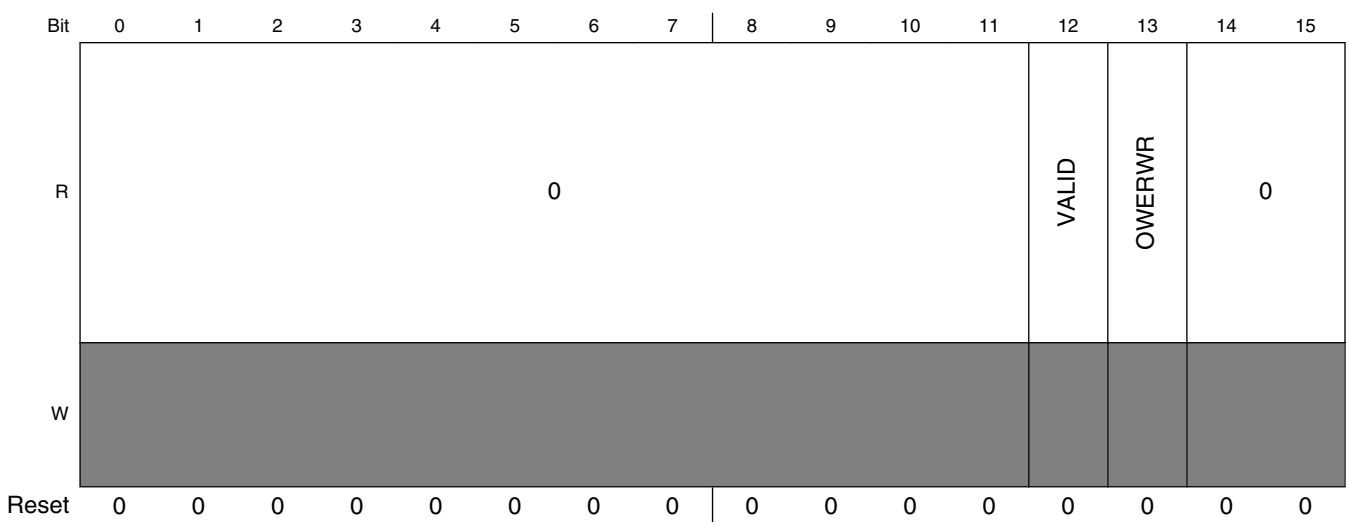
43.5.33 Self Test Data Register 1 (ADC_STDR1)

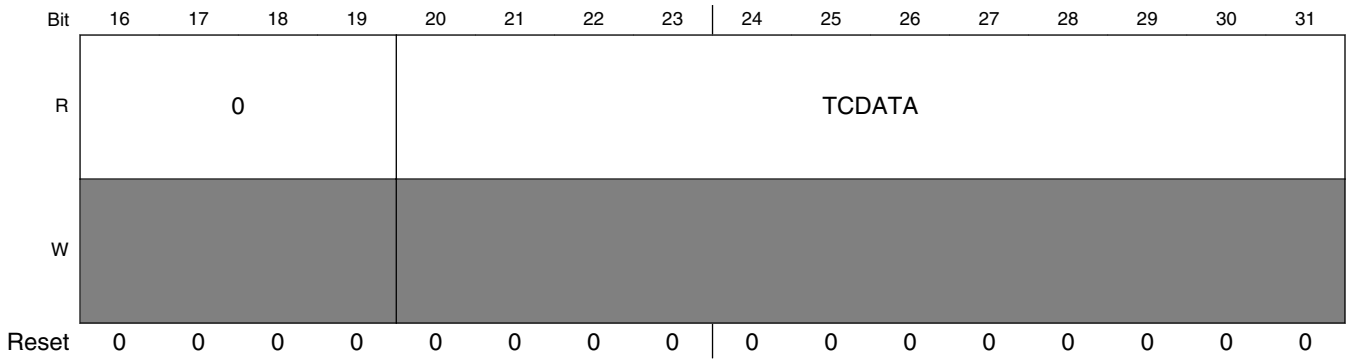
The ADC_STDR1 register contains the most current result data for conversions executed during ADC self test. Additionally, the register contains status bits to indicate whether result data has been read and whether existing conversion data has been overwritten by a newer result.

NOTE

- For Algorithm S step 1, the result data contained in this register is not the final step 1 result. The step 1 data is the conversion of the analog supply (VDDA). This VDDA conversion result is divided by the Algorithm S step 0 conversion data and the result is written to the ADC_STDR2 register.
- The conversion performed in Algorithm S Step 1 is not on the full scale analog supply. Instead, the voltage sampled is VDDA / 8. The pre-scaling is done to avoid issues that can occur in the analog portion of the ADC when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 at the internal stage of data processing before being written to this register to scale the result to the expected level.

Address: 0h base + 370h offset = 370h





ADC_STDR1 field descriptions

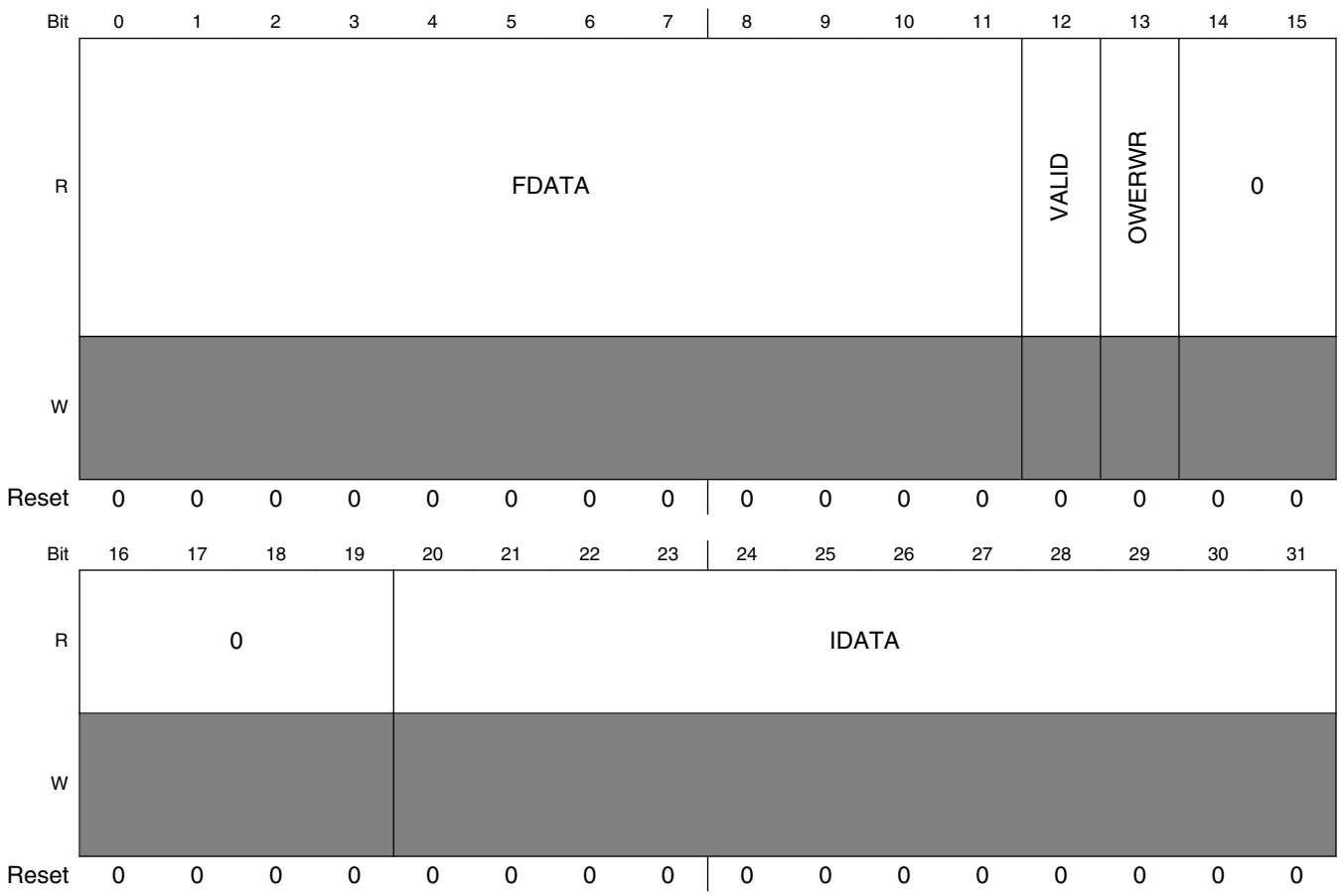
Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data. Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 TCDATA	Test channel converted data. - Unsigned format for Supply Self test - 2's complement for Capacitive Self test

43.5.34 Self Test Data Register 2 (ADC_STDR2)

NOTE

If overflow error occurs for STEP1 of Algorithm-S, STDR2 is not written.

Address: 0h base + 374h offset = 374h



ADC_STDR2 field descriptions

Field	Description
0–11 FDATA	Fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data.

Table continues on the next page...

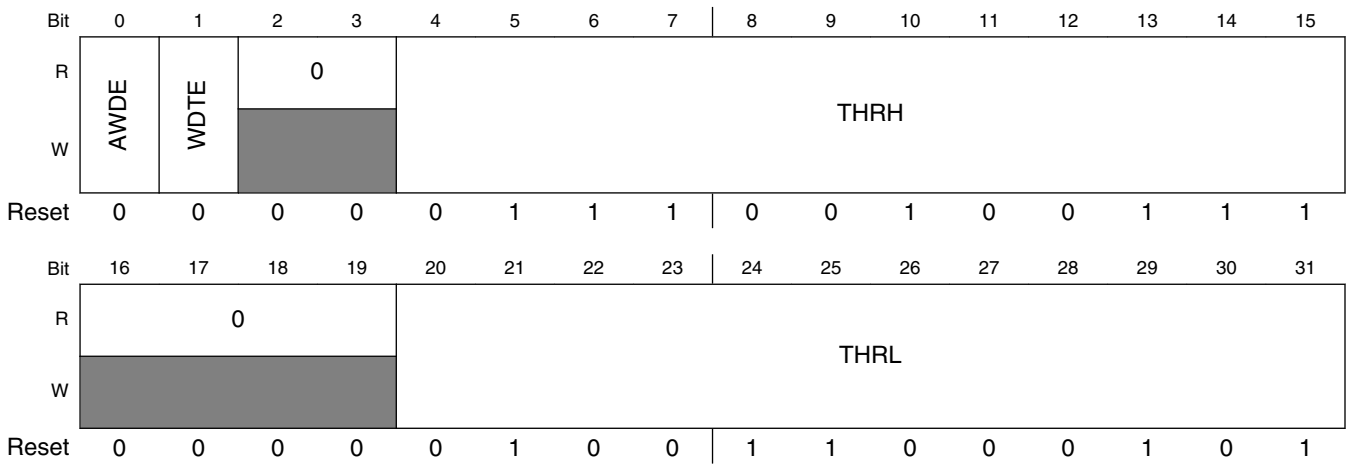
ADC_STDR2 field descriptions (continued)

Field	Description
	Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 IDATA	Integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.

43.5.35 Self Test Analog Watchdog Register 0 (ADC_STAW0R)

This register provides control values associated with the ADC's supply self test step 0, which samples bandgap voltage.

Address: 0h base + 380h offset = 380h



ADC_STAW0R field descriptions

Field	Description
0 AWDE	Analog watchdog enable (related to the algorithm S (step 0)) Enables/disables the comparison of the conversion result from the ADC's supply self test step 0 to the thresholds contained in this register (THRH and THRL) 0 Disabled 1 Enabled
1 WDTE	Watchdog timer enable (related to the algorithm S) Enables/disables the watchdog timer monitoring function for all ADC supply self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT] As soon as the watchdog timer is enabled the algorithm execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode.

Table continues on the next page...

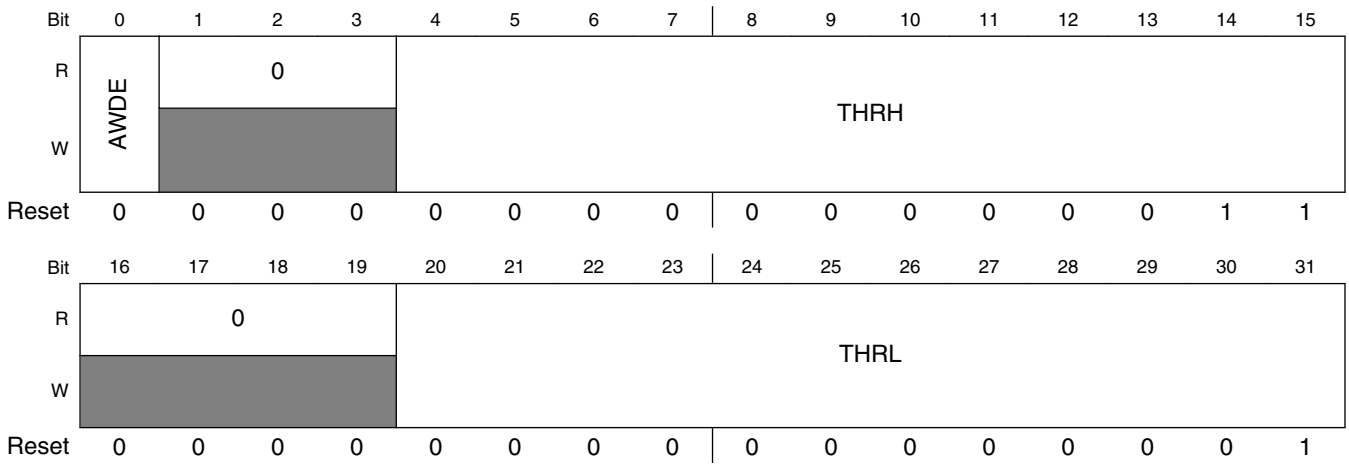
ADC_STAW0R field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) > THRH. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

43.5.36 Self Test Analog Watchdog Register 1A (ADC_STAW1AR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the integer portion of the result. Thresholds in the ADC_STAW1BR register are applied to the fractional portion of the result.

Address: 0h base + 384h offset = 384h



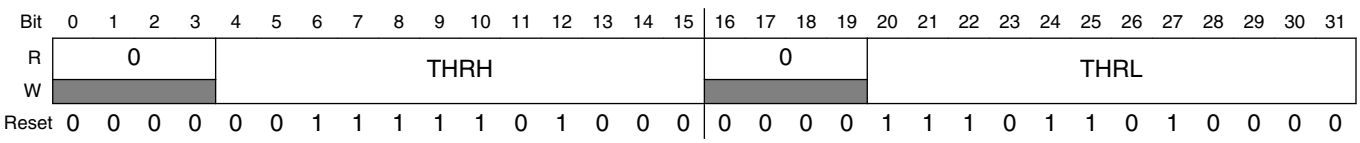
ADC_STAW1AR field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step1).
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

43.5.37 Self Test Analog Watchdog Register 1B (ADC_STAW1BR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the fractional portion of the result. Thresholds in the ADC_STAW1AR register are applied to the integer portion of the result.

Address: 0h base + 388h offset = 388h



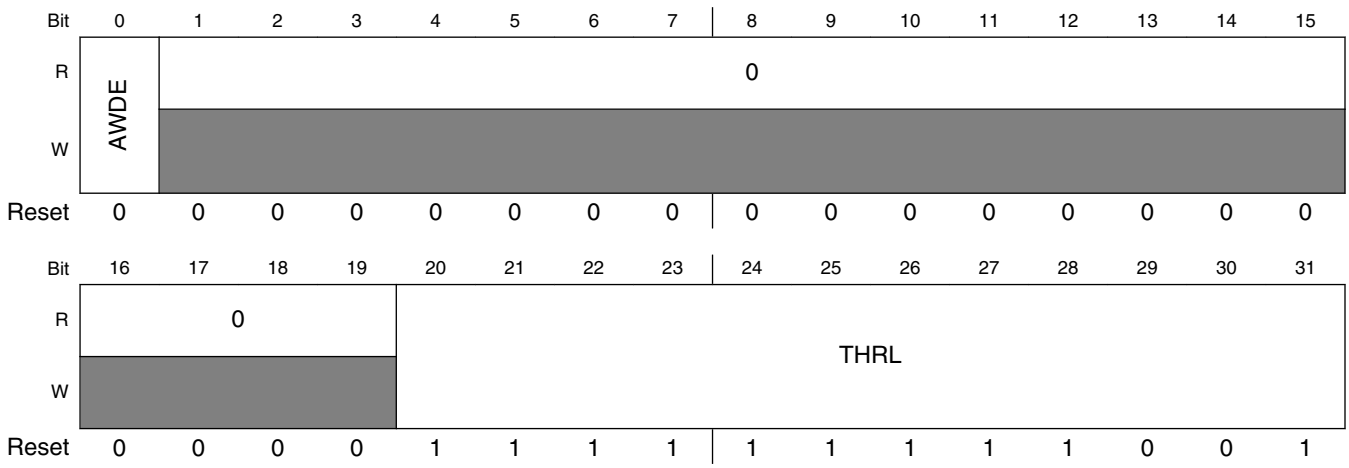
ADC_STAW1BR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (fractional part) for test channel for algorithm S (step 1)(unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (fractional part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

43.5.38 Self Test Analog Watchdog Register 2 (ADC_STAW2R)

This register contains control values associated with the ADC's supply self test step 2, in which the ADC's high reference voltage is sampled and converted. Note that there is no upper threshold value in this register since the conversion result should ideally be FFFh (12-bit) or 3FF (10-bit).

Address: 0h base + 38Ch offset = 38Ch



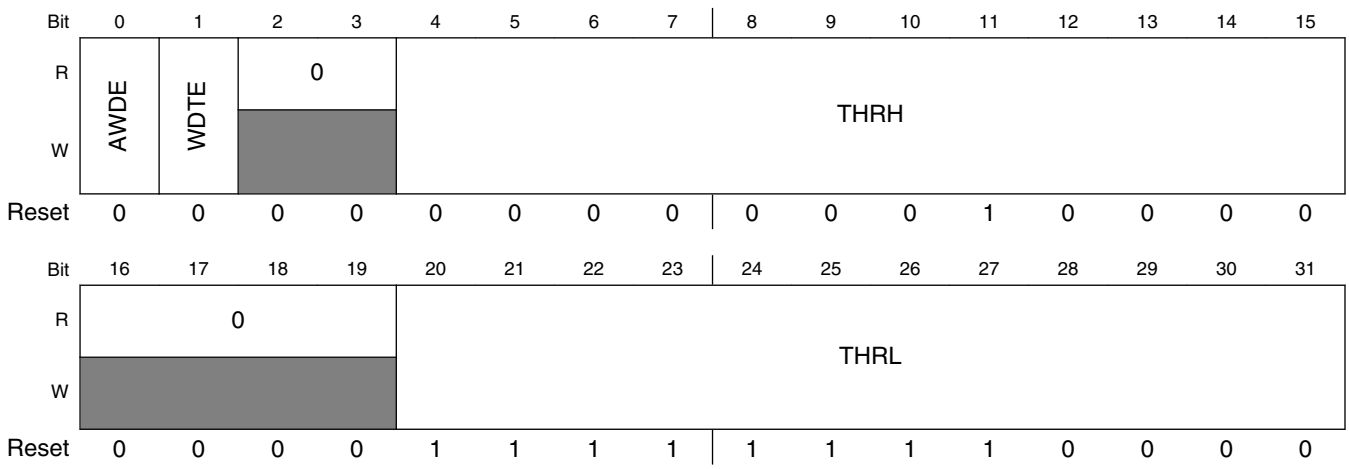
ADC_STAW2R field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step2).
1–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x (unsigned coding). If the analog watchdog is enabled, the STSR1[ERR_S2] status bit is set if STDR1[TCDATA] < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

43.5.39 Self Test Analog Watchdog Register 4 (ADC_STAW4R)

This register contains ADC self test configuration controls associated with the ADC's capacitive self test (also referred to as "Algorithm C") sequence, which samples voltages from the ADC's internal capacitor array. The watchdog enable fields (AWDE and WDTE) affect all Algorithm C steps; the threshold fields (THRH and THRL) fields are specific to Algorithm C Step 0.

Address: 0h base + 394h offset = 394h



ADC_STAW4R field descriptions

Field	Description
0 AWDE	Analog watchdog enable (related to the algorithm C) Enables/disables the analog watchdog monitoring function for all ADC capacitive self test steps. 0 Disabled 1 Enabled
1 WDTE	Watchdog timer enable (related to the algorithm C). Enables/disables the watchdog timer monitoring function for all ADC capacitive self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT] As soon as the watchdog timer is enabled, execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode. 0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step 0 of C algorithm.

Table continues on the next page...

ADC_STAW4R field descriptions (continued)

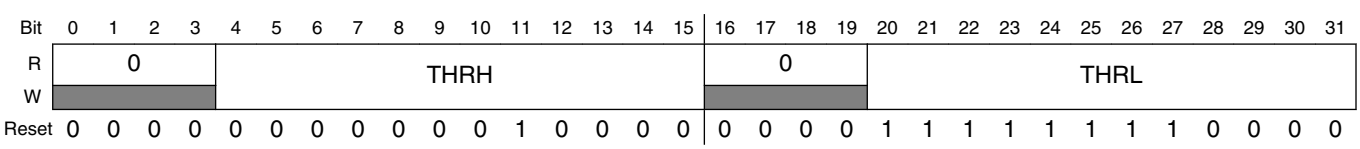
Field	Description
	This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] > STAW4R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW4R[THRH]. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] < STAW4R[THRL], then STSR1[ERR_C] is set. **NOTE:** The settings in this register are applicable only to step 0 of the ADC's capacitive self test. Refer to the ADC_STAW5R register for settings applicable to other capacitive self test steps. The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

43.5.40 Self Test Analog Watchdog Register 5 (ADC_STAW5R)

This register contains ADC self test threshold values associated with all capacitive self test (also referred to as "Algorithm C") steps other than Step 0, which is controlled by fields in the ADC_STAW4R register.

Note that the threshold values in this register are distinctly different than those in the ADC_STAW4R register. The threshold values in this register do not represent a voltage conversion value relative to the ADC high reference voltage, but instead represent the upper and lower thresholds for the absolute difference between the voltage sampled and converted in an Algorithm C step and the voltage sampled and converted in Algorithm C Step 0. Ideally, the difference will be zero.

Address: 0h base + 398h offset = 398h



ADC_STAW5R field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step N of C algorithm (N = 1 to CS-1). This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) > STAW5R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW5R[THRH]. If the watchdog is enabled (STAW4R[AWDE = 1]) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) < STAW5R[THRL], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

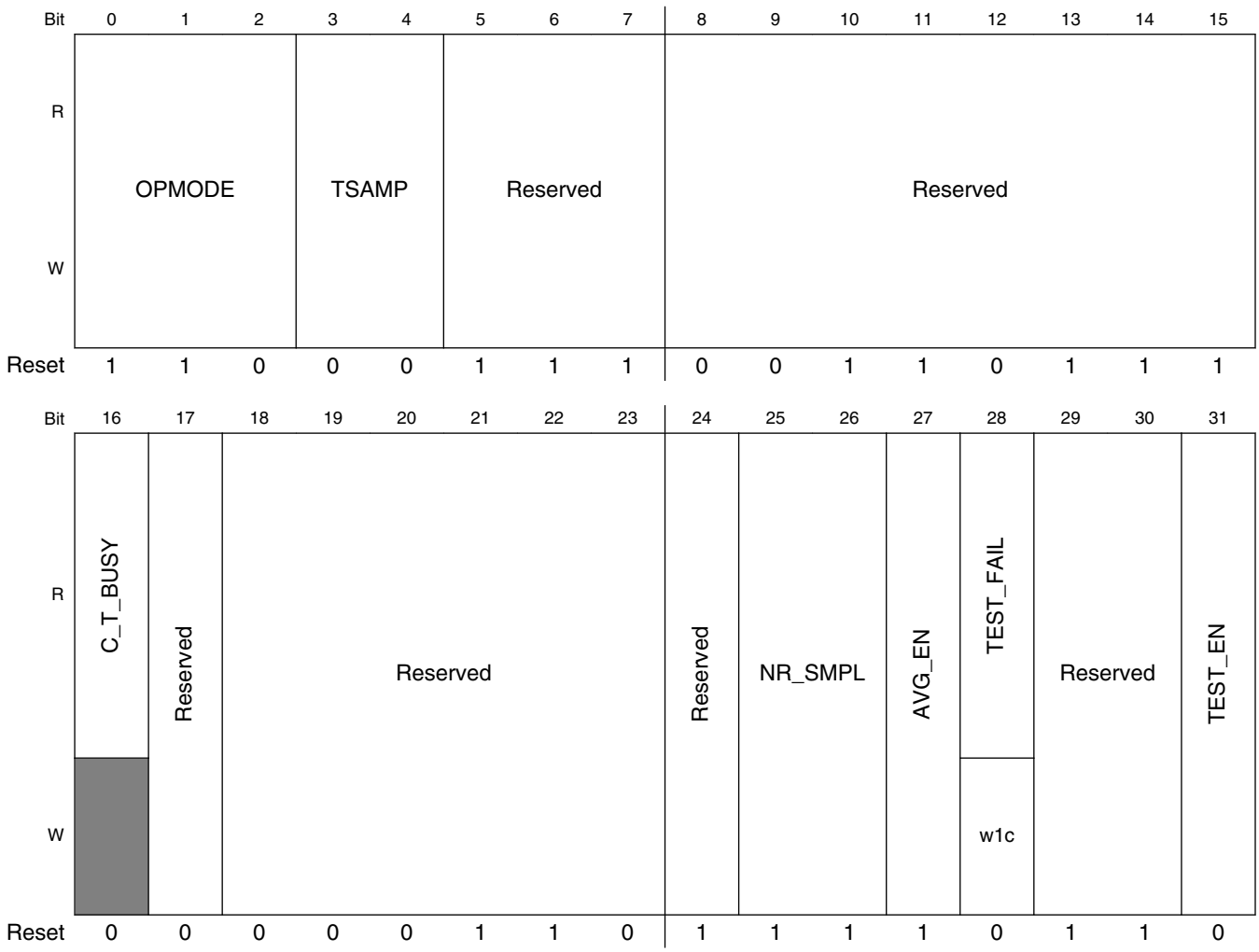
43.5.41 Calibration, BIST Control and status Register (ADC_CALBISTREG)

This register controls the Offset calculation, Calibration and BIST functionality with Averaging option. The averaging function has no effect in other conversion modes.

NOTE

CALBISTREG retains value after all Long Functional Resets except the one caused by an external reset event. It also retains value after Short Functional Resets.

Address: 0h base + 3A0h offset = 3A0h



ADC_CALBISTREG field descriptions

Field	Description
0–2 OPMODE	This field specifies the operating mode of the ADC (normal/high accuracy). The high accuracy mode comes at a cost of 4 additional ADC cycles (80 MHz/40 MHz). In order to keep the same total conversion time as normal mode, the sample time has to be reduced. 000 Reserved 001 12-bit operating mode (normal) 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 12-bit operating mode (high accuracy, recommended) 111 Reserved
3–4 TSAMP	Test Sample period in Calibration, BIST and Offset calculation process. 00 22 cycles of ADC clk 01 8 cycles of ADC clk

Table continues on the next page...

ADC_CALBISTREG field descriptions (continued)

Field	Description
	10 16 cycle of ADC clk 11 32 cycle of ADC clk
5–7 Reserved	This field is reserved.
8–15 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
16 C_T_BUSY	This bit indicates that ADC is doing internal operations for BIST or Calibration. This is set when the TEST_EN bit is set. 0 ADC is ready for use 1 ADC is busy in calibration or test operation
17 Reserved	This field is reserved.
18–23 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
24 Reserved	This field is reserved. The default value must not change. Always write the default value to this field.
25–26 NR_SMPL	Number of Samples for averaging. **NOTE:** The sample setting is providing a trade-off between calibration time and accuracy. For best accuracy 512 samples are recommended. The 16 sample setting shall only be used for applications which require very fast calibration time at the expense of accuracy. 00 16 samples 01 32 samples 10 128 samples 11 512 samples
27 AVG_EN	Average Enable (for Calibration only). 0 Disable 1 Enable
28 TEST_FAIL	Test Fail. This bit indicates the status of Calibration output. This bit sets automatically if tests are aborted prematurely by any normal conversion or tests complete but one of them fails. This bit is cleared by writing 1 to it or on restarting of test by writing 1 to TEST_EN 0 Test passed 1 Test failed
29–30 Reserved	This field is reserved.
31 TEST_EN	Enable the test. 0 Disable the test 1 Enable the test (self clearing) (once set it cannot be overwritten till it resets)

43.5.42 Offset and Gain User Register (ADC_OFSGNUSR)

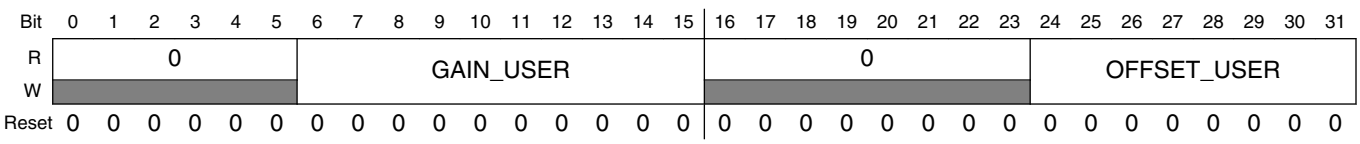
This register contains the user configured offset and Gain values used by ADC for data processing and have all the bits valid (MSB through LSB).

OFFSET and Gain Calibration values can be either positive or negative, so they need to be in 2's complement format with MSB as sign bit.

NOTE

OFSGNUSR retains value after all Long Functional Resets except the one caused by an external reset event. It also retains value after Short Functional Resets.

Address: 0h base + 3A8h offset = 3A8h



ADC_OFSGNUSR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 GAIN_USER	Gain User (2's complement format)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OFFSET_USER	Offset User (2's complement format)

43.6 Functional description

The following sections detail the ADCD interfaces as well as the functionality of this module.

43.6.1 Conversion

After power-up or reset, the ADC is in power-down mode until the MCR[PWDN] field is written. There are some configurations, listed below, available only in power-down mode, that must be handled before exiting power down.

- Clock selection for SAR Controller (AD_clk): MCR [ADCLKSEL]

After the above configurations, ADC need to be taken out from power-down mode by writing '0' into MCR [PWDN], Conversions can be initiated now in the following modes:

- Normal
- Injected
- CTU Triggered

43.6.1.1 Normal conversion

The conversion mode generally used to convert channels is called normal conversion. Programming the available Normal Conversion Mask Register(s) (ADC_NCMR*n*) selects the channel(s). Each channel can be individually enabled by setting a bit in the corresponding field of the NCMR*n* register.

- NCMR0 serves *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)

The mask register(s) must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends. Conversion always starts from the lowest channel number selected and sequentially goes to the highest number.

43.6.1.1.1 Starting a normal conversion

After programming the MCR and NCMR0*n* register(s), a normal conversion is started via *software* by setting the MCR[NSTART] bit.

43.6.1.2 Normal conversion operating modes

Two operating modes are available for normal conversion:

- One-Shot mode
- Scan mode

The MCR[MODE] bit determines which mode is used during a normal conversion. The first phase of the conversion process involves sampling the analog channel. The next phase is the conversion phase, during which the sampled analog value is converted to digital as shown in the figure below.

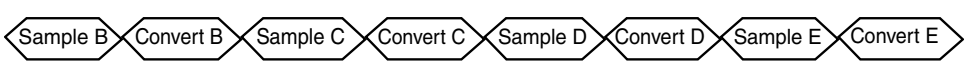


Figure 43-2. Normal conversion flow

43.6.1.2.1 One shot mode

In one shot mode (MCR[MODE] = 0) a sequential conversion specified in the following mask register(s) is performed only once:

- NCMR0

At the end of each conversion, the digital result of the conversion is stored into the corresponding data register(CDRx, x = channel number).

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in One-Shot mode. Conversion starts with channel B, followed by the conversion of channels D and E. At the end of conversion of channel E, the scanning of channels will stop.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time, MCR[NSTART]is reset by hardware, allowing the software to program a new start of conversion in advance. In this case, the newly requested conversion starts after completing the running/current conversion. However, for the sake of simplicity and straightforwardness, it is advisable to program the NSTART bit for the new conversion only after the completion of the running conversion. To know the end of conversion of running chain, application will either wait for an End of Chain (ECH) Interrupt or check the status of MSR[NSTART] prior to setting MCR[NSTART] again.

43.6.1.2.2 Scan mode

In scan mode (MCR[MODE] = 1), a sequential conversion of *n* channels specified in the NCMR*n* register(s) is continuously performed. At the end of each conversion, the converted result is stored into the corresponding data registers, as in One-Shot mode.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One-Shot mode, the MCR[NSTART] bit is not reset automatically in Scan mode. It can be reset by software when you need to stop Scan mode. In that case, the ADC completes the current chain and, after the last conversion, resets MSR[NSTART].

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in Scan mode. MCR[MODE] = 1 is set for Scan mode. Conversion starts from channel B, followed by the conversion of channels D and E. At the end of the conversion of channel E, the conversion of channel B starts again, followed by the conversion of channels D and E. This sequence repeats itself until MCR[NSTART] is reset by software.

This scan chain can be stopped by writing 0 to MCR[NSTART]. The conversion stops when the ongoing chain is finished.

43.6.1.2.3 End of conversion

In both modes, at the end of each conversion an end-of-conversion (EOC) interrupt is issued if enabled by the corresponding mask bit in the following register(s):

- CIMR0

Additionally, the EOC interrupt must enabled in the IMR register. After the conversion of all the channels selected in the NCMR*n* register(s) is completed and the conversion operation is considered finished, ISR[ECH] is set and an end-of-chain (ECH) interrupt is issued (if enabled in IMR[MSKECH]). The corresponding channel bit within the appropriate CEOCFR*n* register is updated to indicate that data is available on the data register (CDR*n*) of the respective channel. If there is no channel selected in the NCMR*n* register(s) and there is a start of conversion trigger, then ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

NOTE

To add or remove channels to/from the ongoing chain:

1. Clear the MCR[NSTART] bit
2. Set/clear mask bit for the needed channels in NCMR*n*
3. Set the MCR[NSTART] bit

43.6.1.3 Injected conversion

An injected conversion is a conversion chain that can be injected at any time, including into an ongoing normal conversion chain. To do so, configure the injected mask register(s):

- JCMR0

As in a normal conversion, each internal or external channel can be individually selected for injected conversion.

- JCMR0 controls *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)

Injected conversion can only be run in One-Shot mode, and can only interrupt a normal conversion. Please note, that a CTU conversion cannot be interrupted by an injected conversion. In this case, the injected conversion is discarded. When an injected chain is inserted into an ongoing normal chain, the ongoing normal channel is suspended and the injected channel request is processed. After completing the last channel in the injected chain, normal conversion resumes from the channel at which it was suspended as shown in the following figure.

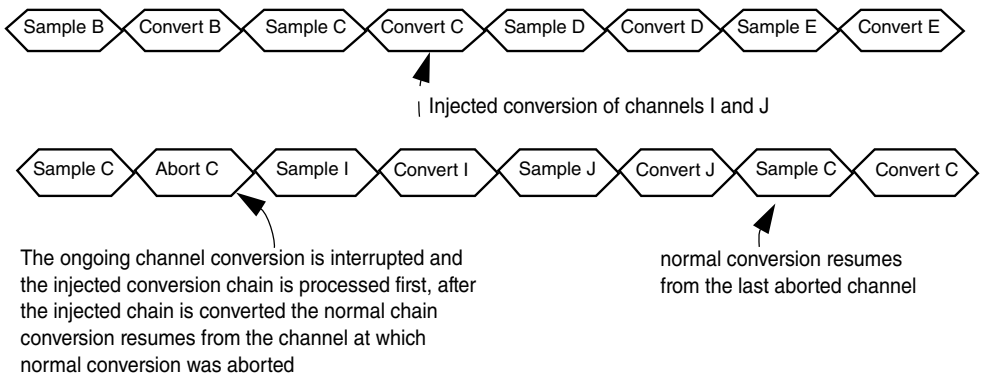


Figure 43-3. Injected sample/conversion sequence

43.6.1.3.1 Starting injected conversion

The injected conversion can be started via *software* by setting the MCR[JSTART] bit to start an injected conversion; the current normal conversion is suspended and the injected chain is converted.

To start an injected conversion via *external trigger*, enable external triggering by setting MCR[JTRGEN]=1. A programmed event (rising/falling edge depending on MCR[JEDGE] bit) on the injection trigger input starts the injected conversion.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time, the MCR[JSTART] bit is reset, allowing software to program a new start of conversion in advance. In that case, the new requested conversion starts after the running conversion is completed.

At the end of each conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit in IMR) and at the end of the chain an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit in IMR). The status bit ISR[JECH] is also set.

If the content of all the injected conversion mask registers (JCMRn) is zero (i.e., no channel is selected) the interrupt JECH is immediately issued to resume normal operation or conversion mode.

The corresponding channel bit in the appropriate CEOCFR*n* register is updated to indicate that data is available in the data register (CDR) of the respective channel.

Once started, injected chain conversion cannot be interrupted by Normal or another Injected conversion. A CTU trigger has the higher priority; injected conversion has a lower priority. Therefore, only a CTU trigger can interrupt an ongoing injected conversion. In this case, the injected conversion will be discarded.

43.6.1.4 Aborting conversions

Two different abort functions are provided:

- A single channel abort
- A chain abort

You can abort the ongoing conversion by setting the MCR[ABORT] bit. The current conversion is aborted and the conversion of the next channel of the chain is immediately started. The abort action may take 1 to 4 cycles of the bus clock to abort the current conversion depending on the state of the conversion. At the start of any conversion and at the end of a particular conversion, when internal state counters are changing, then abort is delayed by a maximum of 3 cycles to take all states in stable condition. Abort also should not be programmed along with and within 3 cycles of programming the MCR[NSTART] or MCR[JSTART]. This may not have a significant impact from software point of view since on any SoC the next write access comes only after a minimum of 3 clock cycles.

NOTE

Programming Abort or Abort chain with NSTART or JSTART in idle condition has no effect.

For a Normal conversion chain having only one channel:

1. Abort re-runs the conversion again from the Abort point

In the case of an abort operation, the MSR[NSTART/JSTART] bit remains set, if not in last channel of chain, and the MCR[ABORT] bit is reset as soon as the channel is aborted. The EOC corresponding to the aborted channel is not generated. This behavior is true for Normal or Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported by generating an ECH interrupt.

It is also possible to abort the current chain of conversions by setting the MCR[ABORTCHAIN] bit. In that case, the behavior of the ADC depends on the MCR[MODE] bit (One-Shot/Scan) for Normal conversion. In One-Shot mode, MSR[NSTART] is automatically reset together with the MCR[ABORTCHAIN]. Otherwise, in Scan mode, a new chain is started, and the current chain is aborted. The EOC of the current aborted conversion is not generated, but an ECH interrupt is generated to signal the end of the chain. For Injected conversion, the behavior of ABORTCHAIN is same as for One-shot Normal conversion. In this case, MSR[JSTART] is automatically reset together with the MCR[ABORTCHAIN].

When an ABORTCHAIN is requested while an injected chain is running over a suspended normal chain, both injected and normal chains are aborted; both MSR[NSTART] and MSR[JSTART] are reset. ABORT and ABORTCHAIN requests are ignored during CTU conversion. ABORT requests are also discarded during Self-Test conversion.

43.6.2 Crosstriggering Unit interface

The Crosstriggering Unit (CTU) interface enhances the injected conversion capability of the ADC. The CTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. The CTU/ADC interface is shown in the figure below. The CTU generates a trigger and a channel number to be converted. A single channel is converted for each request. After performing the conversion, the ADC returns the result.

The conversion result is also saved in the corresponding data register and it is compared with watchdog thresholds if requested. DMA and interrupt feature of ADC can also be used during CTU conversion.

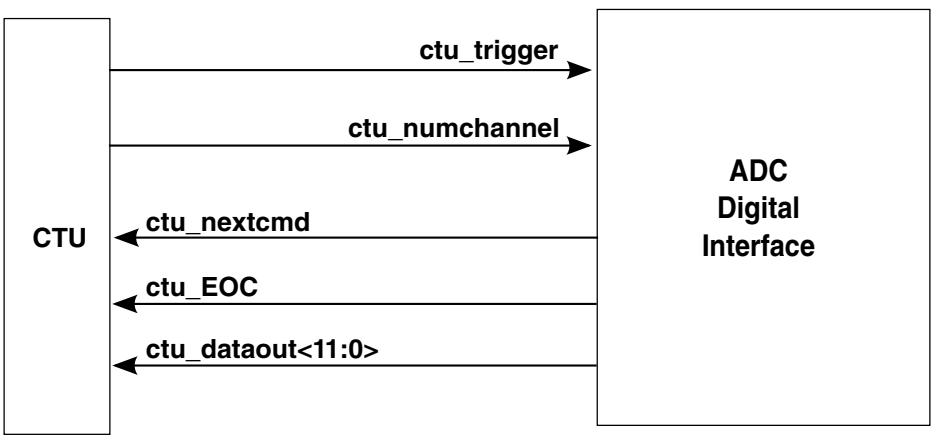


Figure 43-4. ADC Crosstriggering Unit

The CTU interface has one mode of operation, Control.

The CTU interface is enabled by programming the MCR[CTUEN] bit.

43.6.2.1 CTU Control mode

In CTU Control mode, if enabled via MCR[CTUEN], the CPU is able to write in the ADC registers, but it cannot start a conversion. Conversion requests can be generated only by a CTU trigger. If a normal or injected conversion is requested when the CTU is enabled (MCR[CTUEN]), it will be automatically discarded.

Along with CTU trigger the ctu_numchannel is taken as channel and CTU conversion starts. MSR[CTUSTART]is set automatically at this point and it remains set unless the CTU is disabled by resetting MCR[CTUEN].

CTU conversions must be requested when calibration is over. If a CTU trigger is received during calibration, the calibration stops immediately in order to satisfy the CTU request. Calibration will fail in this case.

43.6.3 ADC clock prescaler and sample time settings

The clock (AD_clk) provided to the ADC's SAR controller, which controls the ADCA, must satisfy particular conditions of frequency and duty cycle. The maximum acceptable frequency is 80 MHz with a duty cycle equal to 50% (±5%).

The AD_clk frequency can be scaled by programming the MCR[ADCLKSEL] bit. If this bit is set, AD_clk frequency is the same as the bus clock. Otherwise, AD_clk frequency is half of the bus clock. MCR[ADCLKSEL] can only be written in power-down, when MCR[PWDN] = 1.

Sampling times are controlled by settings in the Conversion Timing Register (CTR*n*).

- CTR0 controls sampling time for *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)

The exception is the Temperature Sensor channel, which always uses the CTR1 value. See Conversion Timing Register 1 (ADC_CTR1) for details.

43.6.4 Presampling

Presampling allows the ADC internal capacitor to precharge or discharge to a defined level before it starts the actual sampling/conversion of the analog input coming from the pads. This is useful for resetting information (history effect/offset) regarding the last converted data. During presampling, the ADC samples the internally generated voltage while, during sampling, the ADC samples the analog input coming from the pads.

Presampling can be enabled/disabled on an individual channel by setting the corresponding bit in the applicable PSR*n* register.

- PSR0 controls precision channels

After enabling the presampling for a channel, the normal sequence of operation is presampling + sampling + evaluation for that channel. Sampling of the channel can be bypassed by setting PSCR[PRECONV]. When sampling of a channel is bypassed, the presampled voltage will be converted.

See Conversion time for the conversion's timing equation.

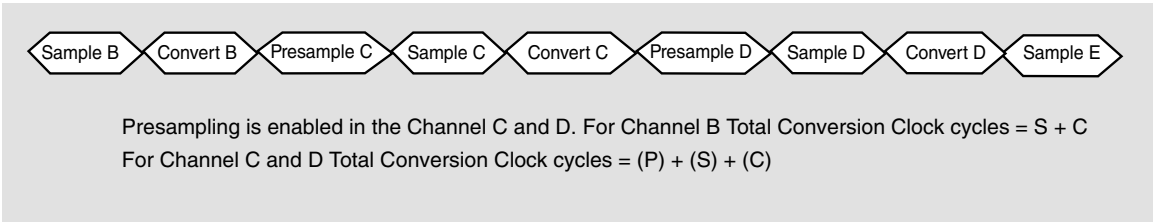


Figure 43-5. Presampling sequence

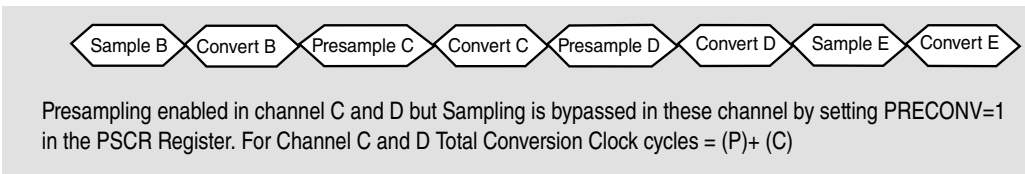


Figure 43-6. Presampling sequence with PRECONV = 1

43.6.4.1 Presampling channel enable

The presampling channels are selected by programming the presampling register (Presampling Control Register (ADC_PSCR)). These are used to enable analog switches that samples an internally generated voltage. It is possible to select presample voltage from four internally available voltages depending on the value of PREVAL*x* [*x* = 0,1,2] fields in the PSCR register as given in Table 43-4.

Table 43-4. Presample voltage selection

PREVAL*n*	Presampling voltage
00b	Presample voltage 1: VSS_HV_ADC (SAR ADC ground)
01b[1]	Presample voltage 2: VDD_HV_ADC/8 (SAR ADC supply divided by 8)
10b	Presample voltage 3: VSS_HV_ADCREF0/1, VSS_HV_ADCREF0/1 (SAR ADC reference ground)
11b	Presample voltage 4: VDD_HV_ADCREF0/1, VDD_HV_ADCREF0/1 (SAR ADC reference supply)

1. The voltage sampled inside ADCA is always (analog supply) / 8 for PSCR[PREVAL*n*] = 01b. However, when sampling is bypassed (PSCR[PRECONV] = 1), ADCD multiplies the conversion result by 8 to make the conversion result equal to the analog supply, or maximum value possible during saturation.

   So, for an open or short circuit test on any channel with presampling feature and PSCR[PRECONV] = 0b, user should expect conversion results as shown below:
   - Equal to Vin of the channel (if there is no open or short defect)
   - Close to (analog supply) / 8 if the channel is open (not connected to any voltage source, internally or externally)
   - Close to 0 V if the channel has a short to ground (analog ground, either internally or externally).
   - Close to the analog supply if the channel has a short to supply (analog supply, either internally or externally).

The PREVAL*n* fields are associated with specific channels, as shown in Table 43-5.

Table 43-5. PREVAL*n* channel association

Field	Channels
PREVAL0	• 0 - 15 (ADC0) • 0 - 15 (ADC1)

The temperature sensor channel is an exception. It can be mapped with any of the channels, but it uses the PREVAL1 value to select the presample voltage because its physical properties may be different than the other internal precision channels.

43.6.5 Programmable analog watchdog

NOTE

The number of analog watchdogs and channels may vary depending on the ADC instance on your chip. Please see the chip-specific section for how many are available on each instance.

The analog watchdogs are used for determining whether the result of a conversion lies within a given guard area (as shown in Figure 43-7) specified by an upper and a lower threshold value named THRH and THRL, respectively.

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside the guard area, then the corresponding threshold violation interrupt is generated.

The comparison result is stored as WTISR[WDGxH] and WTISR[WDGxL], as explained in the following table. Depending on the WTIMR[MSKWDGxL] and WTIMR[MSKWDGxH] mask bits, an interrupt is generated upon threshold violation.

Table 43-6. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

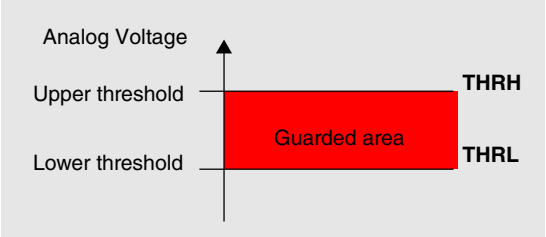


Figure 43-7. Guarded Area

43.6.5.1 Analog watchdog setting

For each watchdog there is one Threshold value (Low and High) Register (THRHLR).

The analog watchdog for each channel can be enabled independently by programming CWENR*n* register bits.

- CWENR0 controls *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)

The availability of these registers depends on device configuration.

Threshold values for each channel can be selected independently from a maximum of 16 threshold registers (THRHLRx,(x=0..15)) by the WSEL_CHx field of CWSELRx (x=0..11) register.

Each CWSELRx register holds selection for 8 consecutive channels. CWSELR0 holds 8 WSEL_CH fields for channel 0 to 7. Likewise CWSELR1 is for channel 8 to 15 and so on. The availability of fields and registers depends on device configuration.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in the Analog Watchdog Out of Range Register (AWORR).

For example, if channel number 12 is to be monitored with the threshold values in register THRHLR3, then CWSELR1[WSEL_CH4] needs to be programmed with 3. The enabling will be done by setting the bit corresponding to channel 12 in the CWENR0 register.

A set of threshold values (THRHLRx) can be linked to several ADC channels. The threshold values to be selected for a channel needs be programmed only once in the CWSELRx register.

43.6.6 DMA functionality

Conversion data of any channel can be transferred from register to system memory via Direct Memory Access (DMA). The DMA transfers can be enabled by setting ADC_DMAE[DMAEN]. Once enabled, the on chip DMA controller can get DMA requests after the conversion of every channel by setting the respective masking bit in the DMAR*n* registers. The DMA masking registers must be programmed before starting any conversion.

The DMA request to DMA controller can be cleared at different times in two modes:

MODE-1: Clearing of DMA request on Acknowledgement from DMA controller (ADC_DMAE [DCLR] = 0)

MODE-2: Clearing of DMA request on Read to data registers (ADC_DMAE [DCLR] = 1)

The figures below shows DMA operation in two modes (cycle counts are typical values).

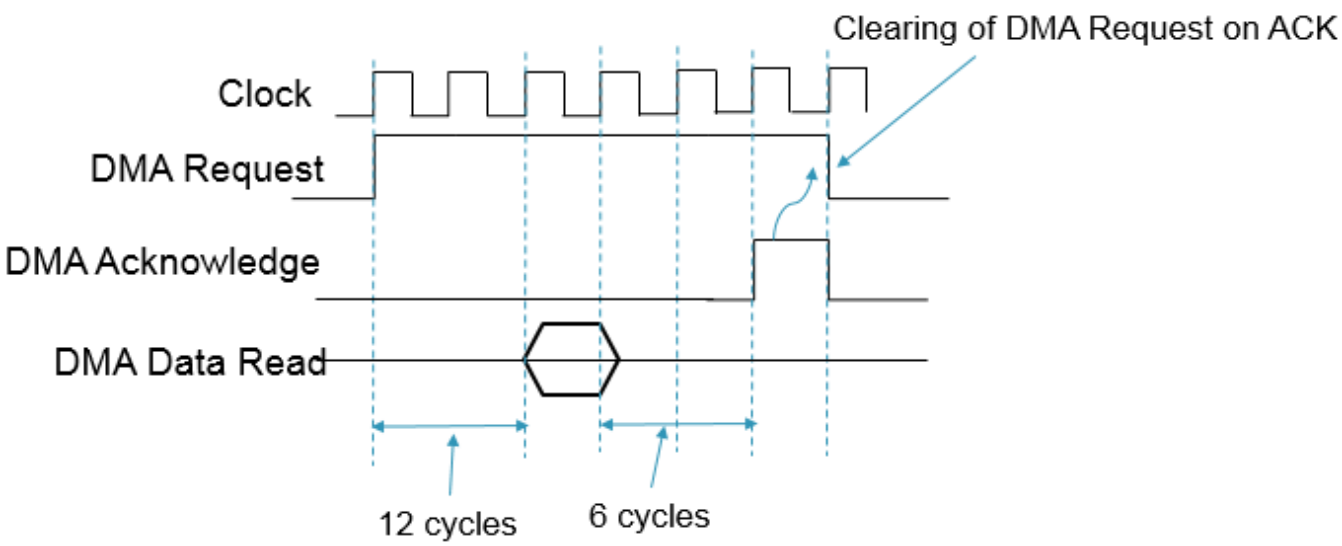


Figure 43-8. DMA operation MODE – 1 (DMAE [DCLR] = 0)

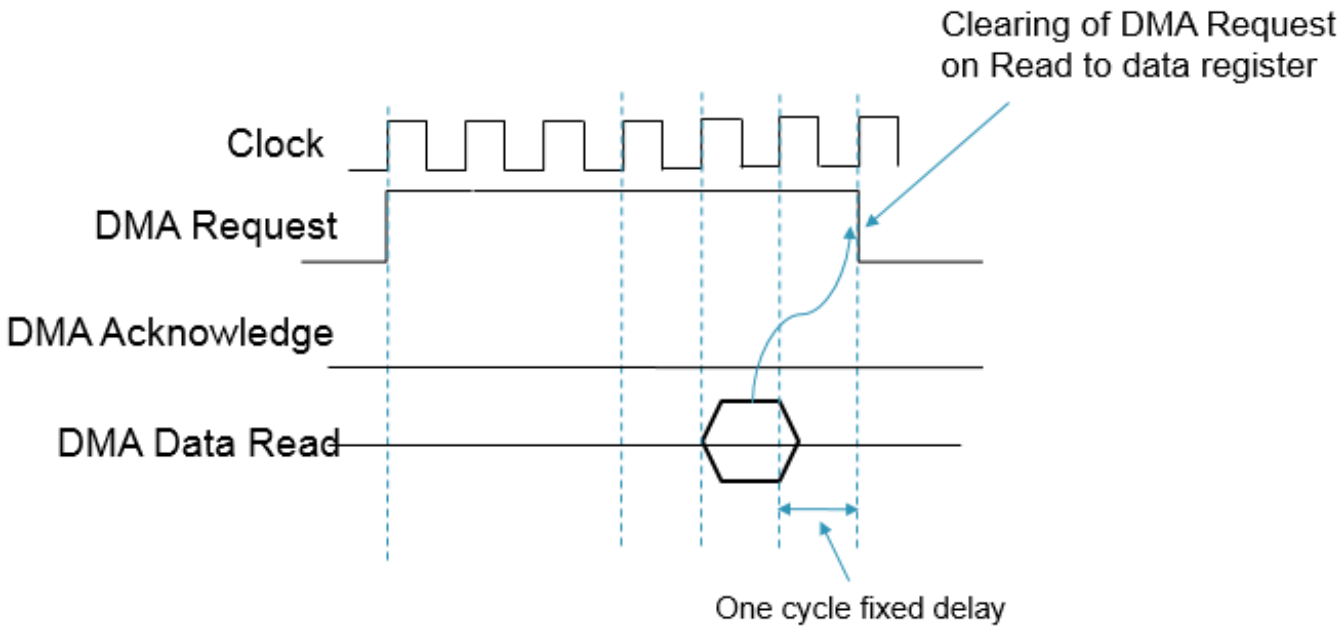


Figure 43-9. DMA operation MODE – 2 (DMAE [DCLR] = 1)

43.6.7 Interrupts

ADC generates the following maskable interrupt signals:

- End-of-conversion interrupt (EOC) request
- End-of-chain interrupt (ECH) request
- End-of-injected conversion interrupt (JEOC) request
- End-of-injected chain interrupt (JECH) request
- End-of-CTU conversion interrupt (EOCTU) request
- Watchdog threshold interrupt (WDGxL and WDGxH) requests

Interrupts are generated during the conversion process to signal events, like the end of a conversion, as explained in the register description for the Interrupt Status Register (ISR). The ISR and the Interrupt Mask Register (IMR) allow you to check and enable the interrupt request.

Interrupts can be individually enabled on a channel-by-channel basis by programming the Channel Interrupt Mask Register (CIMR).

A Channel Pending Register (CEOCFR*n*) is also provided in order to signal which of the channels' measurement has been completed.

Analog Watchdog interrupts are enabled by Watchdog Threshold Interrupt Mask Register (WTIMR) and the status can be checked at Watchdog Threshold Interrupt Status Register (WTISR). The watchdog interrupt source sets two fields, WDGxH and WDGxL, for each channel monitored in the WTISR.

In particular, the EOC, ECH, EOCTU, JEOC, and JECH interrupt lines are combined (OR-ed) on the same line. The EOC interrupt request is cleared by clearing either ISR[EOC] or all bits of CEOCFR. The JEOC interrupt request is cleared by clearing either ISR[JEOC] or all bits of CEOCFR. The EOCTU interrupt request is cleared by clearing either ISR[EOCTU] or all bits of CEOCFR. Also, the self-test end of algorithm interrupts STSR1[WDG_EOA_S] and STSR1[WDG_EOA_C] are combined.

All WDGxH and WDGxL requests are combined on the second line.

Another interrupt on the third line is a combination of WDSERR, WDTERR, ERR_S0, ERR_S1, ERR_S2, and ERR_C in STSR1.

The ISR register contains the interrupt pending request status. If you want to clear a particular interrupt event status, then writing a 1 to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all the other bits of the ISR register must be programmed to 0.)

43.6.8 Power-Down mode

The analog part of the ADC can be put in Power-Down mode by setting the MCR[PWDN] bit, which shuts off the ADCA and stops the clock in the SAR controller. After power-on or device reset, the ADCA is kept in Power-Down mode by default, so this state must be exited before starting any operation by resetting the MCR[PWDN] bit. In Power-Down mode, no conversion and calibration can be started. Trigger from CTU during power down is discarded. In the ADC power-down mode, the CTU must not start any conversion.

The MCR[PWDN] bit may be programmed high at any time. However, if a conversion is ongoing, then the ADCA cannot be sent immediately into Power-down mode. In fact, the ADC enters Power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN function.

The ADC status in MSR[ADCSTATUS] field shows Power-down status only when ADC enters in Power-down mode.

If CTU is enabled and MSR[CTUSTART] is 1, then the MCR[PWDN] bit cannot be set. When CTU Control mode is enabled, the application needs to reset the CTUEN bit before entering Power-Down mode.

After the power-down phase is complete, the process ongoing before the power-down phase must be restarted manually .

NOTE

Resetting the MCR[PWDN] bit and setting the MCR[NSTART] or MCR[JSTART] bit during the same cycle is forbidden.

43.6.9 Auto-Clock-Off mode

To reduce power consumption in Idle mode (without going into Power-Down mode), an auto-clock-off feature can be enabled by setting MCR[ACKO]. When enabled, the analog clock is automatically switched off when no operation is ongoing (that is, when no conversion has been programmed).

NOTE

This feature must remain off during calibration.

43.6.10 Calibration and Self Test

The ADC is calibrated and tested runtime through the same set of test conversions. The test result is used for computation and stored during the calibration process and only checked in self test. The two modes of running tests are as follows:

1. High-Accuracy mode/Calibration. Needs to be run after power-on reset and whenever required in runtime operation. Calibrates the ADC and updates calibration-related registers.
2. Quick-Check mode/Self-Test. Used for runtime functional self-tests to improve reliability. Catches any runtime fault or accuracy shift from previously calibrated values. Please see the chip-specific configuration information as self-test may not be available in all ADC instances.

The following sections describe both and the steps for executing them.

43.6.10.1 Calibration (High Accuracy mode)

The functioning of the analog block of ADC depends on the accuracy of values of its various parts and parameters, such as the value of capacitors, the gain of the amplifier, and so on. Due to variations in manufacturing and various runtime environmental effects, the actual values may differ from the ideal values for which the ADC is targeted.

Because of these deviations, the ADC's converted values contain errors. To eliminate these errors, the ADC must be calibrated prior to any conversion.

In the calibration process, a fixed known reference voltage (VrefH) is sampled many times and converted under various controlled conditions to check for deviations between these converted values and predefined values.

The deviations, known as offsets or modified values, are used to eliminate errors during the data processing of normal conversions.

Note

The ADC must be recalibrated if the operating conditions (particularly VrefH) change significantly. Calibration should also be performed again if the Self-Test indicates that a recalibration is needed.

Recommended settings:

- Clock frequency = 40 MHz
- Averging – Maximum (512)

The calibration process will take ~12ms of time with the above recommended settings.

NOTE

If the ADC clock frequency is > 40MHz during calibration, the accuracy of the calibration process will be compromised, hence the performance of the ADC. Thus, it is strongly recommended to run the calibration with 40 MHz clock.

As calibration is a time consuming process, it can be performed after a long interval – probably several hours of operation – but it is not necessary to recalibrate frequently. The goal is to maximize the possible accuracy delivered from conversion and to catch any structural fault to flag the error.

The calibration routine sets the fail flag (CALBISTREG[TEST_FAIL]) to indicate:

- A fault in ADC
- A premature abort of the calibration

Calibration is aborted if a normal conversion is initiated during calibration; this is called *premature termination*. The terminating conversion and the immediate next one to it will have invalid result in this case.

CALBISTREG[TEST_FAIL] = 1 indicates that the ADC health is not good and that it will not meet specifications.

To execute this routine, follow these steps:

1. Configure the ADC to 40 MHz.
   - Put the ADC in power down mode: set MCR[PWDN]
   - Configure clock divider selection: reset MCR[ADCLKSEL]
   - Power-up the ADC: reset MCR[PWDN]
2. Configure the Calibration, BIST Control, and Status Register (CALBISTREG). The default values are set for maximum accuracy (recommended).
3. Start calibration: set CALBISTREG[TEST_EN].
4. Wait for calibration to complete. Check MSR[ADCSTATUS] or till CALBISTREG[C_T_BUSY] becomes 0.
5. Check the value of MSR[CALIBRTD] to determine whether calibration was successful.
6. If MSR[CALIBRTD] = 1, calibration was successful; otherwise, it failed, and CALBISTREG[TEST_FAIL] = 1.
7. If required, reconfigure the ADC clock frequency for normal operation.

43.6.10.2 Self test (Quick-Check/Safety mode)

For devices used for very critical applications requiring high reliability it is important to check at regular intervals if the ADC is functioning correctly. For this purpose, Self Testing feature (Quick Check) has been incorporated inside ADC.

The following testing algorithms are available for checking:

- Supply Self test: algorithm S. It includes the conversion of the bandgap, supply and VREF voltages. It includes a sequence of three test conversions (steps). The supply test conversions must be an atomic operation (no functional conversions interleaved).
- Capacitive Self test: algorithm C. It includes a sequence of 12 test conversions (steps) to check the capacitive array.

The capacitive test of ADC requires only 12 steps to test itself.

The capacitive array is used as DA converter to generate from V_{REFH} 12 different voltages by distributing and partially discharging portions of the capacity array. These 12 voltages are sampled using some redundant capacities. C algorithm may be compared to providing 12 different voltages to ADC and validate the conversion result.

Two kinds of tests are performed in this mode, as shown in the following table.

Table 43-7. Quick-Check mode tests

STCR3 [ALG]	STCR3 [MSTEP]	Description	Outcome	Comment
00 supply test	0	Supply self-test (band gap voltage)	Measures the internal band gap voltage	—
	1	Supply self-test (analog supply)	Measures the analog supply voltage (VDDA) and divides the result by the bandgap voltage (result from Algorithm S step 0). The VDDA conversion result is written to ADC_STDR1[TCDATA]. The final result (after division) consists of an integer portion and a fractional portion, both of which are written to the ADC_STDR2 register.	The conversion performed in Algorithm S Step 1 is not the full scale analog supply. Instead, the sampled voltage is VDDA / 8. Prescaling is done to avoid issues that can occur in the analog portion of the ADC when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 in the digital part of the ADC to scale the result to the expected level.
	2	Supply self-test (reference voltage high)	Measures the high reference voltage of the ADC	—
01	Reserved	—	—	—
10 capacitive test	0–11	One of the ADC-BIST steps is being run	The difference or error of individual value from the previously calibrated value is being returned (unsigned) as an ADC result that is compared with a programmed value in the analog watchdog register(s) to detect any fault. The difference or error may be caused by any runtime fault, runtime accuracy shift, or device noise.	Ideally, the result value is expected to be 0. If the returned value is higher, this indicates runtime fault or accuracy shift. This indicates that ADC should be recalibrated in High-Accuracy mode to check whether the reported error can be handled by recalibrating or whether permanent damage to ADC has occurred.
11	Supply test followed by Capacitive test in same sequence as shown above	—	—	—

This mode is described in the following sections.

43.6.10.2.1 Self test initialization and application information

ADC samples the internal V_{REFH} for the C algorithm and samples different supplies (bandgap, V_{DD} and V_{REFH}) for the supply algorithm. It also provides signals for the following:

- Scheduling self-testing algorithms using configuration registers

- Monitoring the converted data using analog watchdog registers
- Flagging the error to FCCU in case a failure occurs in any of the algorithms

ADC mode	TEST channel (ADCdig)	TEST channel (CTU)
CPU mode (MCR[CTUEN] = 0)	Yes • One-Shot mode • Scan mode	No
CTU Control mode	No	Yes

43.6.10.2.2 CPU mode

In this case, the test channel works similarly to a normal conversion. The test channel is enabled by setting STCR2[EN].

Self-test conversions are executed by interleaving the Normal conversions. The sequencing of steps of the selected algorithm for test channel depends on the operating mode of normal conversion, selected by MCR[MODE].

In One-Shot mode, if the test channel is enabled, only one step of the selected self-testing algorithm is executed at the end of the chain. The step number and algorithm to be executed are programmed in the STCR3 register. So, in One-Shot mode, the sequence is as follows:

1. Program NCMR*n* register(s) to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 0 to select One-Shot mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. The default is algorithm S.
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of all normal conversions, the step number programmed in STCR3[MSTEP] of the self-testing algorithm selected by STCR3[ALG] is executed similar to a normal functional channel.
10. On receiving an end-of-conversion for the test channel, the digital result is written in STDR1[TCDATA] andSTDR1[VALID] is set. Also, ISR[EOC], ISR[ECH], and STSR1[ST_EOC] are set. See the register descriptions (Self Test Data Register 1 (ADC_STDR1), Interrupt Status Register (ADC_ISR), and Self Test Status Register 1 (ADC_STSR1)) for details of settings for each test.
11. The state machine returns to the Idle state.

For example: in a device with channels A-B-C-D-E-F-G-H, channels B-D-E are to be converted in One-Shot mode. MODE = 0 is set for One-Shot mode. Refer to Conversion. Conversion for channels B-D-E are done. After channel E test channel conversion is done and ISR[ECH] and ISR[EOC] are set.

MSR[NSTART] is automatically set when the normal conversion starts; it is reset at the end of conversion for the test channel.

In Scan mode, the consecutive steps of the selected self test algorithm are converted continuously at the end of each chain of normal conversions. The number of channels converted at the end of each chain is 1 (except for algorithm S, in which all the steps are performed at once without any functional conversion interleaved). So, in Scan mode the sequence is as follows:

1. Program the NCMR*n* registers to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 1 to select Scan mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected (that is, all algorithms are executed step-by-step one after the other).
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of chain for the normal conversions (assuming a default value of STCR3[ALG]), all steps of algorithm S are performed (as algorithm S is always atomic). MSR[SELF_TEST_S] is set.
10. At the end of conversion of test channel for the last step of algorithm S, the digital result is written in STDR1[TCDATA] and STDR1[VALID] is set. At the same time, MSR[SELF_TEST_S] is reset. For step 1, the integral part and fractional part are written in STDR2[IDATA] and STDR2[FDATA]. Also, ISR[EOC], ISR[ECH], and STSR2[ST_EOC] are set.
11. The next normal conversion chain starts.
12. At the end of the normal conversion chain, Step0 of the C algorithm is executed.
13. At the end of conversion of test channel for Step0 of the C algorithm, the digital result is stored in STDR1[TCDATA] and STDR1[VALID] is set (if MCR[OVERWR] is set). Also, ISR[EOC], ISR[ECH], andSTSR2[ST_EOC] are set.
14. The next normal conversion chain starts.
15. At the end of the normal conversion chain, step1 of C algorithm is executed.
16. This process continues for all steps of all algorithms.
17. The state machine returns to the Idle state when MCR[NSTART] is written to 0.

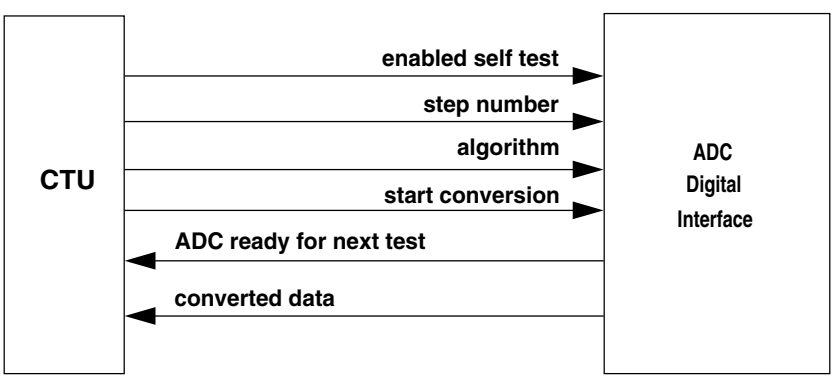
NOTE

- Test channel conversion is not performed during injected conversions. It is performed only during normal conversions.
- If, during a test channel conversion, injection conversion arrives, then the test channel is aborted (just as a normal functional channel) and injected conversions are done. After injected conversions are completed, the test channel resumes from the step at which it was aborted. In this case, MSR.SELF_TEST_S remains high during the injected conversion.
- For self testing, MCR[MODE]should be programmed at least one cycle *before* setting MCR[NSTART]. It should not be changed thereafter until a conversion is ongoing.

43.6.10.2.3 CTU mode

For this chip, the operating mode is CTU Control mode.

When MCR[CTUEN] is set, the test channel conversion can be started only by the CTU interface; software cannot start it. STCR2[EN] does not have any effect in CTU mode. The CTUEN bit should not be changed while normal conversion is ongoing. The interface between the CTU and ADCD (for self test) is shown here:



For self testing conversions in CTU mode, the CTU asserts an indication that enables self test along with the CTU trigger that starts the self test conversion. The algorithm and the step number to be executed is put on the interface respectively.

NOTE

As already mentioned for algorithm S, the three steps must be atomic. In CTU mode, this is managed by the CTU itself; that is, the CTU has to send three triggers (one for each step)

asserting the indication that enables self test and updating the step number for each step.

43.6.10.2.4 Abort and Abort Chain for self-test channel

Setting MCR[ABORT] during self-test channel has no effect.

In One-Shot mode, if MCR[ABORTCHAIN] is set when the test channel conversion is ongoing, the test channel is aborted and ECH is set. In this case, EOC for the test channel is not generated.

In Scan mode, if MCR[ABORTCHAIN] is set when test channel STEP N is ongoing, then the test channel STEP N is aborted and the next chain conversion starts. At the end of this chain, STEP N conversion is performed again. (If algorithm S is ongoing, the full algorithm is executed again.)

43.6.10.2.5 Self-test analog watchdog

ADCD also provides a monitor for the values returned by the ADC analog macro for self-test algorithms. The analog watchdogs are used to determine whether the result of conversion for self-test algorithms lie in a particular guard area. For this purpose, separate Self-Test Analog Watchdog Registers are provided for each algorithm.

If the analog watchdog feature is enabled (STAWxR[AWDE] = 1), a comparison is performed between the converted value and the threshold values after the conversion of each step of an algorithm. If the converted value does not lie between the upper and lower threshold values specified by the Analog Watchdog Register of the particular algorithm, the corresponding error bit (STSR1[ERR_x]) is set and the step number during which the error occurred is updated in STSR1[STEP_x] (in case of the C algorithm) and erroneous data is written in STSR4[DATAx]. STSR1[ERR_x] generates an interrupt if enabled by the corresponding mask bit in STCR2. The fault indication is also given to the FCCU via the Critical and Non-Critical fault lines, so that necessary action can be taken at chip level.

The analog watchdog feature works differently in the case of algorithm S. As already mentioned, algorithm S is always an atomic operation. So, separate error bits are provided in STSR1 for each step of algorithm S to avoid an overwrite in case an error occurs in more than one step. Hence, there are separate mask bits for each step in STCR1. For the same reason, separate fields in the status registers (STSR2 and STSR3) exist to store erroneous data for each step.

In step 0 of supply algorithm, ADC measures the band gap voltage (+1.2V), which is assumed to be stable with very little variation (up to +/- 1.5%). The converted data of step 0 is measured with high (THRH) and low (THRL) threshold defined in STAW0R if enabled (STAW0R[AWDE]). The STSR1[ERR_S0] is set to '1' if threshold is violated; once set, it is cleared to '0' by writing '1' to it.

In step 1 of supply algorithm, ADC measures the Analog supply (Vdda) for the ADC, and then a fixed point division is performed with the result of step 1 (ADC supply voltage) and step 0 (band gap voltage). It takes around 26 cycles after EOC for step 1 to get a divided value, and it is the divided value on which analog watchdog checks are applied. So, the value to be compared for step 1 contains the integer as well as fractional part; two registers (STAW1AR and STAW1BR) are provided for this. The comparison is done first for the integer part using the threshold values programmed in STAW1AR. If the integer part lies in the range, the fractional part comparison is skipped; otherwise, it is compared with the values programmed in STAW1BR. The following table summarizes this feature for step 1.

Table 43-8. Algorithm S (step 1) threshold comparison

STDR2[IDATA] (integer part)	STDR2[FDATA] (fractional part)	STSR1[ERR_S1]
> STAW1AR[THRH]	Any value	Set
< STAW1AR[THRL]	Any value	Set
== STAW1AR[THRH]	> STAW1BR.THRH	Set
== STAW1AR[THRL]	< STAW1BR.THRL	Set

In step 2 of supply algorithm, ADC measures the High Reference voltage (Vrh); (VREF/VREF) is measured in order to check the integrity of sampling signal. For this particular conversion, no higher threshold value is required as the ideal value is 0xFFF. Only the lower threshold value is programmed in STAW2R.

In the supply self-test algorithm, the band gap is measured with regard to the reference voltage (Vrh) in step 0. The result is capable of catching mismatches between them. Step 1 measures the supply with regard to reference (Vrh), and the result is taken to do a ratio (Vdda/Vbg). If the band gap voltage remains within +/-1.25% of its ideal value of 1.2 V, then by setting appropriate values of THRH and THRL in watchdog registers, the ADC self-test can catch more than 4% variation in supply and reference faithfully.

Algorithm C has total 12 steps. Output data of step 0 is compared with the threshold values stored in STAW4R, if enabled in the same register. The result of step 0 is used as an offset for rest of the steps (step 1 to 11). The absolute difference between the converted data of each of the steps (1 to 11) with the data of step 0 are compared with the threshold values of STAW5R, if enabled in STAW4R. Error in any step is indicated by STSR1[ERR_C] and the data in error case is captured in STSR4.

Calculation of Threshold values:

S1 Algorithm: Calculated voltage comparison method:

The user software can implement a method of calculating S1 algorithm IDATA, FDATA, the THRH values, and the THRL values to voltages and then perform a comparison of the calculated voltage to the calculated THRH voltage and calculated THRL voltage.

The steps involved in this method are as follows:

1. Calculate the high threshold voltage:

   (STAW1AR [THRH] + (STAW1BR [THRH] / 4096)) * Vbandgap

2. Calculate the low threshold voltage:

   (STAW1AR [THRL] + (STAW1BR [THRL] / 4096)) * Vbandgap

3. Calculate the converted results to voltage:

   (STDR2 [IDATA] + (STDR2 [FDATA] / 4096)) * Vbandgap

4. Compare result from step 3 to THRH and THRL.S1 algorithm passes

   If (low threshold voltage <= converted voltage <= high threshold voltage).

43.6.10.2.6 Watchdog timer

The watchdog timer is an additional check that monitors the sequence of the self-testing algorithm implemented and checks that the algorithm completes within a safe time period. Watchdog timers can be enabled for CPU as well as CTU conversions. Each algorithm has a different watchdog timer that runs independently of the other. The watchdog timer for a particular algorithm can be enabled by setting STAWxR[WDTE]. The safe time value can be programmed in STBRR[WDT] (the default value is 10 ms, assuming an 80 MHz clock).

The safe time is measured starting from step 0 of the algorithm (including all normal chain conversions in between) to the point at which step 0 of the same algorithm starts again.

The sequence is as follows:

1. Program NCMR0 to select channels to be converted for normal conversion in Scan mode (MODE = 1).
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (14 h) (for 1 MSPS operation)

3. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected; that is, all algorithms will be executed step-by-step, one after the other.
4. Enable the self-testing channel by setting STCR2[EN].
5. Program the safe period value in STBRR[WDT].
6. Enable the watchdog timer by setting STAWxR[WDTE]. Assume the setting of WDTE to be t0. (It is important to do all the programming first and to then enable WDTE as the safe time check is also performed between the setting of WDTE and the start of step 0 to check that algorithm has started within the safe time).
7. Start the normal conversion by setting MCR[NSTART].
8. After first chain conversion ends, step 0 of algorithm S is executed. Lets assume the start of STEP0 to be t1.
9. Step 1 and step 2 of algorithm S are executed.
10. The next chain conversions are performed.
11. When chain conversion completes, step 0 of C algorithm is performed.
12. After each chain conversion, a consecutive step of the C algorithm is performed. A similar sequence follows for algorithm C.
13. After the last step of the C algorithm is performed, another chain conversion is executed. At the end of this chain conversion, step 0 of algorithm S starts, repeating the whole sequence. Let's assume this time (starting of step 0) to be t2.
14. For algorithm S, if (t1 – t0) > Safe Period, or if (t2 – t1) > Safe Period, then the watchdog timer flags an error and STSR1[WDTERR] is set. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDTERR]. Otherwise, the watchdog timer counter is reset and starts again to monitor the same for the next sequence.
15. A similar sequence is followed for watchdog timers for algorithm C.

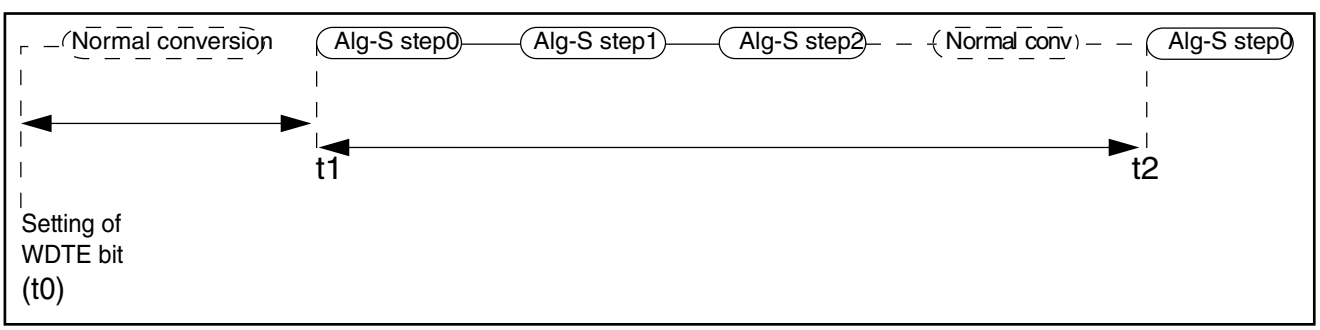


Figure 43-10. Watchdog timer monitor for algorithm S

Notes:

- As CTU does not incorporate any safe period checking mechanism, the watchdog timers can also be enabled for CTU conversions.
- Take care not to enable the watchdog timer for an algorithm that is not to be executed.

43.6.10.2.6.1 Watchdog sequence checking

The watchdog timer also incorporates sequence checking features that check the order of a particular algorithm's steps. If the steps are not in order, then an error is flagged by setting the STSR1[WDSERR]. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDSERR].

A watchdog sequence error is flagged in the following cases:

- If the steps of any algorithm are not executed in the proper order.
- If an abort chain occurs during a test channel conversion, that step has to be repeated at the end of the next chain. This will give a sequence error as soon as test channel conversion starts again.
  - Exception: If an abort chain occurs during the last step of algorithm S, then sequence errors are not flagged as the whole algorithm has to be repeated again.
- If, for CTU conversions, the step numbers provided by the CTU are not in order. Watchdog sequence checking is significant for CTU burst mode only.

If injected conversion occurs during the test channel, watchdog sequence errors are *not* flagged although the ongoing step number is aborted and repeated again.

Note

The watchdog timer feature is applicable only for Scan mode, not for One-Shot mode.

43.6.10.2.7 Baud rate control for test channel

It defines the scheduling of the test channel between normal conversions. The scheduling rate is specified by STBRR[BR].

By default, if the test channel is enabled, one step of the selected algorithm is executed after every chain of normal conversion. The bandwidth consumed by the test channel depends on the number of channels in a normal chain. For example, if there are 100 normal conversions in a chain, then the test channel consumes only 1% of the total bandwidth, which is very small. However, if the number decreases to just four channels, then the bandwidth consumed by the test channel is 25%, which is significant (and may not be desirable as it slows down the normal conversion rate).

STBRR[BR] provides flexibility by scheduling the test channel conversion to be performed, not at the end of every chain, but at the end of BR + 1 number of chains. For example, if BR = 5, a single step of the selected algorithm for the test channel is performed after six chain conversions, then the next step is performed at end of the next six chain conversions, and so on. By default, the value of BR is 0.

Notes:

- This feature is applicable only for Scan mode, not for One-Shot mode. STBRR[BR] should be set to 0 for one shot mode.
- To use the baud rate feature in Scan mode, the NCMR register should have nonzero value.

43.6.10.2.7.1 Abort chain when baud rate is non-zero

As previously described, a nonzero value of STBRR[BR] means that the test channel conversion is performed at the end of BR + 1 number of chains. If an abort chain occurs during the chain in which the test channel is scheduled to be converted, then the test channel is converted after the next BR + 1 number of chains.

For example, if STBRR[BR] is programmed to 2, the sequence will be two normal chains (without any test channel conversion) followed by a chain with the test channel converted at the end. Now, if an abort chain occurs during first two chains it is treated as a normal chain abort and the test channel is converted at the end of the third chain only (as the case without any abort chain). But if the abort chain occurs during the third chain in which the test channel is scheduled to be converted, then the test channel is converted after next three chains (that is, at the end of sixth chain, counting from the beginning).

43.6.11 Conversion time

Total conversion time depends on ADC-SAR controller's operating clock. This clock can be prescaled to bus clock ÷ 2. The ADC-SAR operating clock (AD_clk) is always synchronous with bus clock. The frequency of AD_clk is dependent on settings of ADC_MCR[ADCLKSEL].

If ADC_MCR[ADCLKSEL] = 0, then AD_clk = (bus clock ÷ 2)

If ADC_MCR[ADCLKSEL] = 1, then AD_clk = bus clock

The various components of conversion time and affecting configurations are shown and described in Figure 43-11.

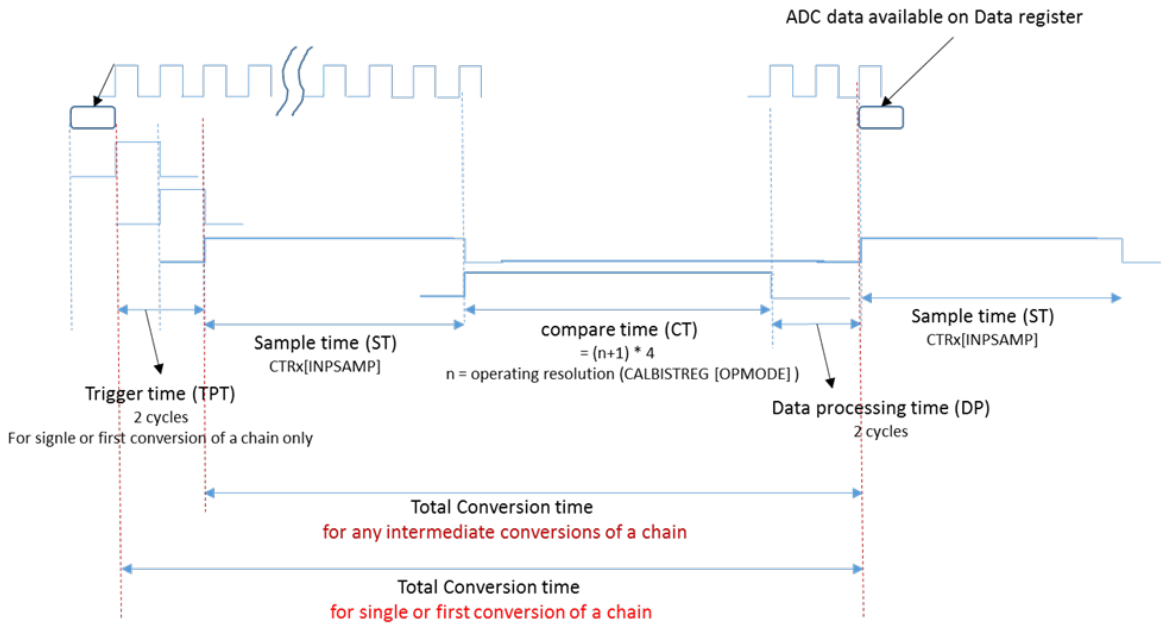


Figure 43-11. Conversion time

- Trigger processing time (TPT):

  To prepare the channel and start the operation, two clock cycles of bus clock is required by the ADC for the first conversion and for subsequent conversions in a chain, or for a continuous conversion, this time is not required as it is hidden in the pipeline operation. Triggers from Synchronous CTU interface requires two cycle of bus clock for first conversion and for back-to-back CTU conversions it takes only one cycle for trigger processing as the other cycle is hidden in pipeline operation. Triggers from Asynchronous CTU interface requires three cycles of bus clock for synchronization and processing.
- Sample phase time (ST):

  Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTR*x* (*x* = 0...2) for different types of channels. Value in each register affects the sample time in similar way. The value in the register represents units of cycle of the AD_clk. The minimum value of sample time is 8. If the value programmed is less than 8, then it has no effect on sample time duration and in this case sample time will be 8 AD_clk cycles.
- Compare phase time (CT):

  The compare phase of the ADC is dependent on the resolution setting. It takes $((n + 1) \times 4)$ cycles of AD_clk, where n is the resolution setting configured in CALBISTREG[OPMODE].

- Data processing time (DP):

The ADC takes 2 cycles of AD_clk to post process and load the data int registers.

- Total Conversion Time:

  The total time required for single or first conversion is [TPT+ST+CT+DP] cycles of AD_clk and for subsequent conversion is a chain or continuous conversion is [ST+CT+DP] cycles of AD_clk.

  Examples:

  - CALBISTREG[OPMODE] = 110b; CTR*x*[INPSAMP] = 20d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 20 + ((13 + 1) × 4) + 2 = 20 + 56 + 2 = 78 cycles of AD_clk
  - CALBISTREG[OPMODE] = 001b; CTR*x*[INPSAMP] = 22d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 22 + ((12 + 1) × 4) + 2 = 2 + 22 + 52 + 2 = 78 cycles of AD_clk
    - Subsequent/continuous conversion: 22 + ((12 + 1) × 4) + 2 = 22 + 52 + 2 = 76 cycles of AD_clk
  - CALBISTREG[OPMODE] = 110b; CTR*x*[INPSAMP] = 20d; CTU triggered:
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 1 + 20 + ((13 + 1) × 4) + 2 = 1 + 20 + 56 + 2 = 79 cycles of AD_clk
  - CALBISTREG[OPMODE] = 001b; CTR*x*[INPSAMP] = 22d; CTU triggered:
    - Single/First Conversion: 2 + 22 + ((12 + 1) × 4) + 2 = 2 + 22 + 52 + 2 = 78 cycles of AD_clk
    - Subsequent/continuous conversion: 1 + 22 + ((12 + 1) × 4) + 2 = 1 + 22 + 52 + 2 = 77 cycles of AD_clk

- Pre-Sampling:

  When pre-sampling is enabled, the ADC does pre-sampling prior to starting the actual channel sampling as described in pre-sampling section. This phase takes CTR*x*[INPSAMP] cycles of AD_clk. Additionally, to switch from pre-sample phase to actual channel sampling phase, it takes another two cycles of AD_clk. So, total CTR*x*[INPSAMP] + 2 cycles additionally required for a conversion to be completed. Thus, in case of pre-sample, all the equations and examples above are valid with the addition of (CTR*x*[INPSAMP] + 2) cycles.

43.6.11.1 Conversion time for CTU interface

The CTU works in the same clock domain of the ADC operating clock allowing the conversion time to be similar to normal conversion time previously discussed.

43.6.12 Conversion data processing

The raw converted ADC data contains many types of errors such as offset, gain, DC bias, and so on. Thus, to generate error-free results, raw converted data is processed before it is written to a result register. The process of error correction goes bit-by-bit during conversion with the values generated during the offset calculation and calibration process. This helps in reducing final data processing time as only one addition/addition (2's complement) is performed simultaneously with conversion.

43.7 Clock frequency

The ADC internal blocks are designed to work with an 80 MHz clock. Thus, the specified speed can be met with this clock frequency. If the clock is faster than this, ADCA cannot work properly and the ADC may malfunction. If the clock is slower, then the ADC speed will be lower, but it will work properly. It should be noted that there is an absolute maximum value of the sampling time that the analog portion of the ADC can support. For slower clocks, the sampling clock count will need to be programmed accordingly.

CTU, which talks to the ADC digital part, may run in different clock domain. For such usages, a clock domain synchronizer is placed at the interface of ADC digital and CTU. The loss is of 2-extra clock cycles due to synchronization; thus overall speed need to be achieved by reduced cycles for samples.

Chapter 44
Temperature Sensor (TSENS)

44.1 Introduction

The device includes two on-chip temperature sensors. The temperature sensor module monitors device temperature and delivers one analog output signal and two digital output signals each (under- and over-temperature flags). The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device (see the ADC details for the channel number). The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage; sampled bandgap reference voltage, which comes from another module; and calibration parameter values stored in internal flash memory. The two digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions.

The digital outputs signal when the junction temperature drifts below the low temperature threshold or above the high temperature threshold. These signals notify the device to take action to appropriately adjust the device temperature in response to an out-of-specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

44.2 Functional Description

The temperature sensor generates one analog output voltage, $V_{TSENS}(T)$, which is proportional to the absolute current junction temperature of the device, and two digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or a pre-set high temperature threshold.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap reference voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

44.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. Refer to "Software-controlled temperature sensor trim adjustment" in Chip-specific power management controller (PMC) information.

When the temperature threshold detection feature is enabled, the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the following temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (-40°C).
- The high temperature digital output signals if the junction temperature rises above the high temperature threshold (150°C).

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40°C), and the high temperature threshold is trimmed to match the hot insertion test temperature (nominally equal to 150°C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on Figure 44-1.

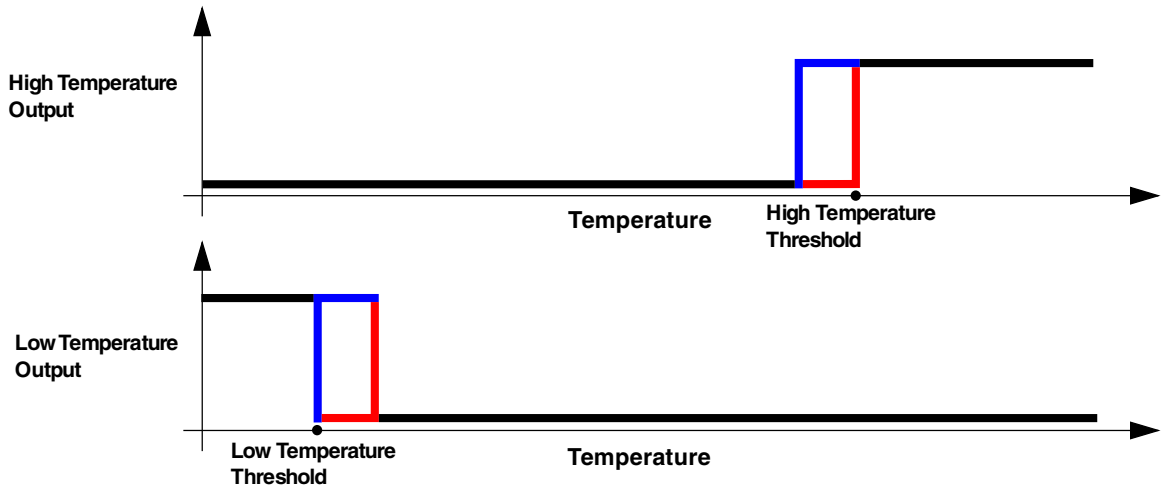


Figure 44-1. Digital output behavior with temperature

44.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

44.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is composed of the temperature sensor, a bandgap reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the bandgap reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in Figure 44-2 is used to calculate internal device junction temperature.

$$T \times 2^2 = \frac{(K_1 \times V_{BG\_CODE}(T)) \times 2^{-1} + (K_2 \times V_{TSENS\_CODE}(T)) \times 2^3}{[(K_3 \times V_{BG\_CODE}(T)) \times 2^2 + (K_4 \times V_{TSENS\_CODE}(T))] \times 2^{-10}} \quad [K]$$

$$V_{TSENS\_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12} \qquad V_{BG\_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

- T is the internal junction temperature in Kelvin
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap reference voltage sampled by the ADC
- V_{ref} is the ADC reference voltage in normal usage of the device
- K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation stored in flash memory (obtained from factory calibration)

Figure 44-2. Temperature formula

Parameters K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation that are stored in flash memory during factory test and calibration. These constants must be used in the temperature formula to calculate device junction temperature.

44.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap reference voltage using the on-chip ADC module and calculate $V_{BG\_CODE}(T)$ according to the formula shown in Figure 44-2.
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $V_{TSENS\_CODE}(T)$ according to the formula shown in Figure 44-2
3. Retrieve parameters K1, K2, K3, and K4 from flash memory.
4. Calculate T according to the formula shown in Figure 44-2.

Note

Temperature Formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

Chapter 45
Signal Processing Toolbox (SPT)

45.1 Chip-specific SPT information

The following table gives the defined sample rates for SPT-MIPICSI2 interactions as per the 1 Gbps limitation from MIPICSI2 and 20 MSPS limitation from SPT.

Table 45-1. SPT-MIPICSI2 sample rate

Data Type	Data Mode	5th channel	Sampla rate per chanel (MSPS)	Sampla rate 5th chanel (MSPS)
RAW8/10/12	REAL (A-D)	OFF	2.5 - 20	N.A.
RAW14	REAL (A-D)	OFF	2.5 - 1 8	N.A.
RAW8/10/12	COMPLEX (A-H)	OFF	2.5 - 10	N.A.
RAW14	COMPLEX (A-H)	OFF	2.5 - 9	N.A.
RAW8/10/12	REAL (A-D)	ON	10	10
RAW8/10/12/14	REAL (A-D)	ON	5	10
RAW8/10/12/14	REAL (A-D)	ON	2.5	10
RAW8/10/12/14	REAL (A-D)	ON	6.67	13.33
RAW8/10/12	COMPLEX (A-H)	ON	5	10

NOTE

The ‘CPU VERSION Registers’ mentioned in Interrupts, corresponds to ‘System Version Register’ of Z7 Cores in this device.

45.1.1 Time Period of calculation of Throughput in various modes of PDMA

The Table 45-73 uses the following time periods for its calculations.

In this device

- spt_clock_period is the time period of SPT_CLK.

- ahb_clock_period is the time period of SYS_CLK.
- ipg_clock_period is the time period of PBRIDGE_n_CLK.

45.2 Introduction

The Signal Processing Toolbox (SPT) contains all the hardware modules required for processing of the sampled RADAR signals. It is a powerful processing engine containing high-performance signal processing operations, driven by a specific use-oriented instruction set. The programmability ensures flexibility for modifications of signal processing. The CPU is removed from frequent scheduling of hardware operations, but still controls and interacts with the processing flow.

45.3 Block diagram

The SPT is connected to the device by an advanced high performance master bus and a peripheral bus.

The system bus master interface performs fast data transfers between external memory and local RAM.

The purpose of the peripheral interface is to set configuration, get status information and basic control of SPT (start/stop, program pointer) and to trigger interrupts. It can also be used to exchange small amounts of the data between the CPU and the SPT, such as constant operands.

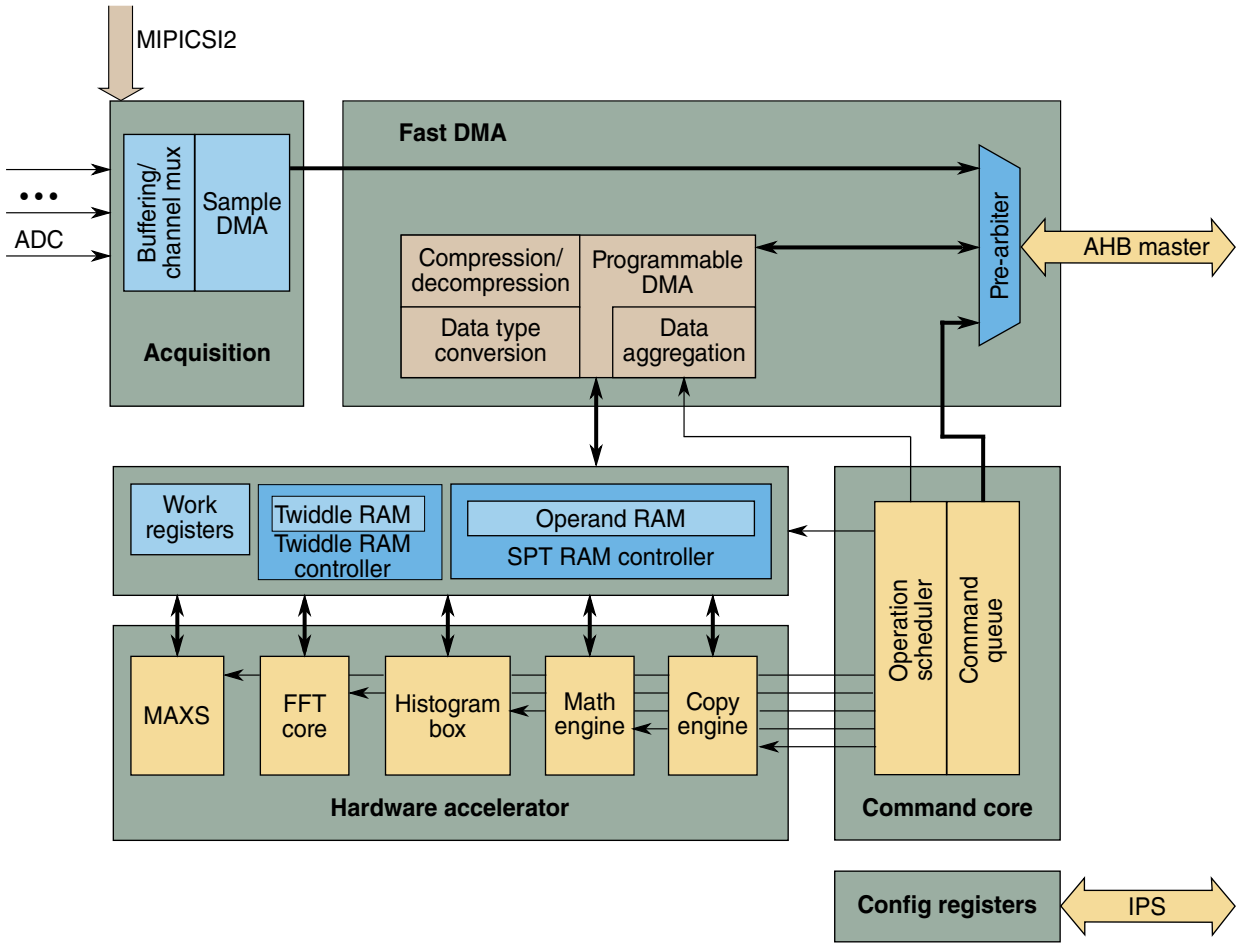


Figure 45-1. SPT Block Diagram

45.4 Features

- Acquisition of ADC samples
  - Capturing of ADC samples within programmed window
  - On-the fly statistical computation
  - Supports MIPICSI2 interface
- Provides HW acceleration for
  - FFT (8 - 4096 point)
  - Histogram calculation
  - Maximum and peak search
  - Mathematical operations on vector data
- High-speed DMA data transfer
  - Supports system RAM, TCM and Flash memory transfers
  - Includes compression/decompression capability for reduction of storage footprint
- Instruction based program flow

    - High-level commands for signal processing operations
    - Simple control commands
    - Local instruction buffer
    - Automatic instruction fetch from main memory
    - CPU interaction possible
- CPU interrupt notification
- Watchdog
- Debug Support - Single Stepping and Jamming Mode

45.5 SPT modes of operation

The following sections provide information about the modes in which SPT module can be used.

45.5.1 STOP Mode

When STOP mode request arrives, the command sequencer goes to the ASYNCSTOP state. The current SDMA, PDMA, or CSDMA burst is completed and reset is applied to SPT clock domain modules (which includes all modules except ACQ, SDMA and DMA arbiter). Also, the DMA arbiter does not request any further bursts on the System bus (though the acquisition module continues to receive data from the ADC). At this point acknowledge to this stop request is given. Thereafter ADC, SPT and AHB clock are stopped.

While exiting from the STOP mode, for proper operation, the acquisition module shall be restarted by resetting and then again setting the ACQ_EN signal. After that, the command sequencer can be restarted by setting the SPT_GBL_CTRL[PG_ST_CTRL] bit. Note that for proper sync after STOP mode exit, all the modules interacting with SPT, like CTE, WGM, MIPICSI2 receiver and transmitter, etc need to be restarted as well.

45.5.2 System Debug mode

When system enters debug mode and requests SPT to go to debug mode then the command sequencer goes to the debug state only after the current instruction has completed execution. This debug mode is mainly for the CPU, and it works in conjunction with another field SPT_CS_MODE_CTRL[CS_SYS_DEBUG]. If this field is zero then the command sequencer ignores the system debug entry request, else it goes to the debug state.

Also note that since the sequencer enters the debug state only at the end of the current instruction, the CPU must wait till the present instruction is completed before it tries to debug errors inside the SPT.

Note that ACQ and SDMA continue working in debug mode.

45.5.3 Normal Mode

The detailed functioning of this mode is provided in the Functional Description section.

45.6 SPT Operation Model

The SPT executes a list of instructions. These commands contain all the information to perform signal processing operations on data vectors consisting of a set of numbers. The instruction list is provided by the CPU in a memory buffer (SRAM or flash) and read by the SPT with autonomous triggered DMA operations. Preparation of command scripts is performed off-line during development on a different system, such as a PC.

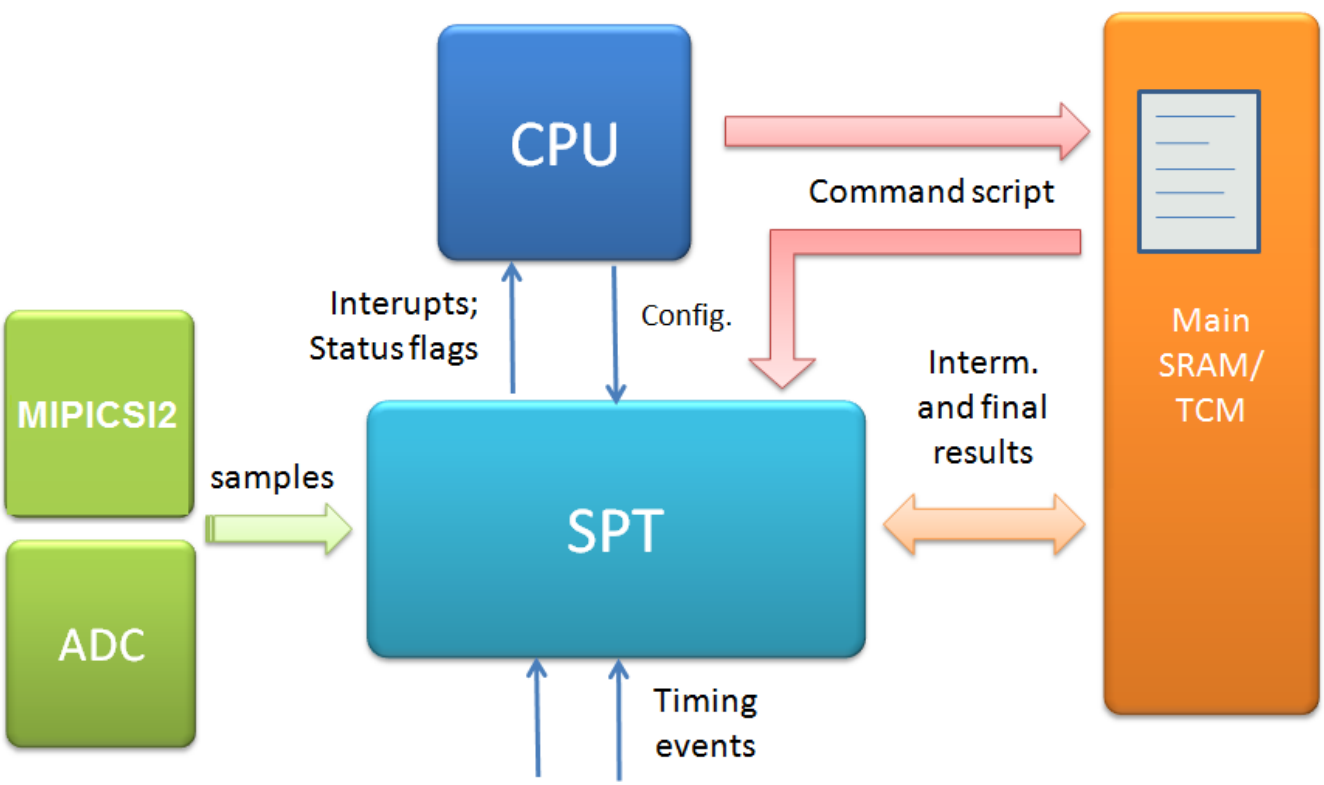


Figure 45-2. SPT Operation Model

Data as results or inputs of signal processing operations can be transferred between operand/twiddle RAM and system RAM or TCM by DMA operations. These kind of DMA operations are also scheduled by commands in the script.

CPU application software functions may be invoked between signal processing operations. In order to synchronize these SW functions with the processing flow, interrupts and polling flags are provided, which may be activated as a result of specific commands. In the same way, eDMA operations can be triggered from the SPT. The data transfer descriptors need to be prepared by application software.

To enable advanced and precisely timed pipelining of HW assisted operations, SPT command execution can also be synchronized with or triggered by events from CTE or MIPICSI2.

The SPT also performs initial, on-the-fly processing for ADC samples to generate statistical results in advance and compensate offsets.

45.6.1 Program Execution

Program execution is performed by a dedicated register machine. The program execution may be stalled at the end of each command and can be synchronized with events provided by the CTE or MIPICSI2. The CPU can stop and release program execution and execute its own instructions in the mean time.

The SPT command sequence is provided by CPU SW and written to a buffer in the system RAM or flash, where CSDMA fetches or alternatively transfers the list of instructions to the command queue.

The program is executed sequentially, one command following the next in the chain with the exception being loops and asynchronous PDMA instruction. The end of a sequence is marked with a specific termination command. The start address of the instruction sequence is configured by application software using the peripheral interface.

The command queue stores a number of instructions to be executed, which may be a shorter sequence than the complete instruction list. It maintains an instruction pointer to the current executed operation, which is incremented when the current operation is completed. This instruction pointer indicates the position in the complete instruction list sequence relative to the first instruction.

Loop instructions are special cases, the program sequence may continue with a lower address depending on the state of the loop counter. The start and next command address of the loops are saved in special registers. Up to four nested loops are supported.

Jump instructions or branch instructions are another case where the program may cause the program to branch to a non-contiguous address location. It is the responsibility of the assembler to ensure that the sanity of the program structure is not violated i.e. infinite loops, jumping into or out of a loop, etc.

The commands are stored in the system RAM or flash in a big endian format i.e. MSB in lower memory byte and LSB in higher memory byte. Table 45-2 shows the organization of a command word in the memory.

Table 45-2. Command word organization in memory

Memory Address	Instruction Bit Range
0x0	[127:120]
0x1	[119:112]
0x2	[111:104]
0x3	[103:96]
0x4	[95:88]
0x5	[87:80]
0x6	[79:72]
0x7	[71:64]
0x8	[63:56]
0x9	[55:48]
0xa	[47:40]
0xb	[39:32]
0xc	[31:24]
0xd	[23:16]
0xe	[15:8]
0xf	[7:0]

45.6.2 Instruction Fetch

SPT instructions are fetched from system RAM when the command queue is empty at the start of a program or when the last instruction in the queue starts during program execution, or even when a command close to the last instruction is being executed and the remaining commands are not loop-end commands. The command queue is filled by means of a SPT DMA operation. The instruction fetcher also handles loads from lower than the current addresses, in case a loop command branches to a lower address or an instruction load from a higher or lower address is required when a jump instruction causes program flow to go to a non-contiguous address. A specific branch prediction for performance increase is not performed. It is assumed that an execution stall due to instruction fetch only takes a few bus cycles and only happens when the longest command chain in a loop does not fit in the queue.

45.6.3 Watchdog

Some signal processing operations may have a limited time budget. A watchdog is available to monitor the processing time of a sequence of commands and raise a flag when a command sequence has taken too long.

The watchdog is armed by an SPT command configuring the initial timer value it can also stop or reset the watchdog. The watchdog works in two modes, count mode and event mode. In count mode the timer counts down using the SPT clock till it reaches zero. In the event mode the watchdog can be reset to zero with any CTE or MIPICSI2 event signal. Hence, in event mode the CTE or MIPICSI2 events can serve as a timing reference.

The current watchdog value can be read by application software. An interrupt may be issued to the CPU if the watchdog counter reaches zero in count mode. Alternatively the application software may read the watchdog value to identify if exceeding the timing budget, which is the case if read value is zero.

45.7 SPT internal memory map

SPT contains internal memory resources namely, Operand RAM, Twiddle RAM and work registers. Out of these, only the work registers can be directly accessed via the peripheral interface and are visible on the SPT external memory map.

The CPU can access these internal memory resources via the hardware accelerators. The instruction bitfields SRC_ADD and DEST_ADD point to these memories.

The two MSB bits of the address define the memory type. Table 45-3 shows the mapping of these address bits to the different memory resources in SPT. The figure below shows the memory map view.

Table 45-3. Memory type address decoding

Address bit value	Memory type	Comments
00b	Work registers	48 Work registers of word width of 48 bits each
01b	Twiddle RAM	512 x 8 locations of 36 bits each
10b	Operand RAM	512 x 8 x 4 locations of 52 bits each
11b	Reserved	Future Operand RAM expansion - Address wrapped around to point to Operand RAM

The Twiddle RAM extends from location 0x4000 to 0x4FFF. However the area from 0x5000 to 0x7FFF is aliased to the Twiddle RAM i.e. it wraps around to point to the Twiddle RAM. It should be noted that the memory accesses to the Twiddle RAM should not cross the area from 0x4000 to 0x7FFF.

The Operand RAM extends from location 0x8000 to 0xBFFF. But the area from 0xC000 to 0xFFFF is aliased to the Operand RAM. The memory accesses to the Operand RAM should not cross the area from 0x8000 to 0xFFFF.

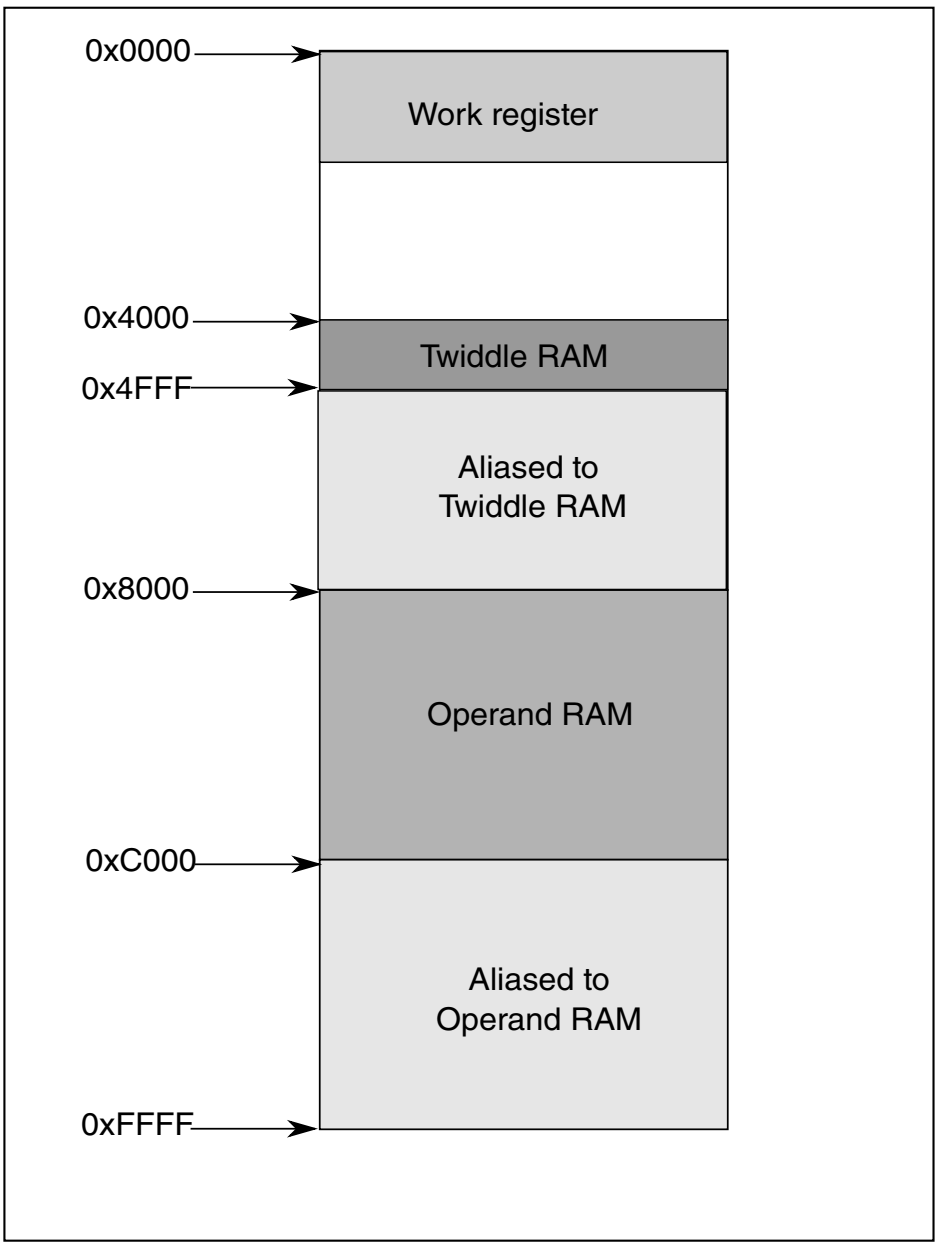


Figure 45-3. Memory Map view

45.8 Memory map and register description

NOTE

Transfer error will not be generated at offset 0x3E0, 0x3E4, 0x3E8 and 0x3EC

The following table shows the SPT detailed memory map:

SPT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SPT Global Control Register (SPT_GBL_CTRL)	32	R/W	0000_0008h	45.8.1/1783
4	Acquisition Global Control Register (SPT_ACQ_GBL_CTRL_0)	32	R/W	0000_0000h	45.8.2/1784
8	Acquisition Global Control Register 1 (SPT_ACQ_GBL_CTRL_1)	32	R/W	001F_001Fh	45.8.3/1787
10	Acquisition Control 0 Register (SPT_ACQ_CTRL0)	32	R/W	0000_0000h	45.8.4/1788
14	Acquisition Control 1 Register (SPT_ACQ_CTRL1)	32	R/W	0000_0000h	45.8.5/1788
18	Acquisition Control 2 Register (SPT_ACQ_CTRL2)	32	R/W	0000_0000h	45.8.6/1789
1C	Acquisition Control 3 Register (SPT_ACQ_CTRL3)	32	R/W	0000_0000h	45.8.7/1790
20	SDMA Control 0 Register (SPT_SDMA_CTRL0)	32	R/W	0000_0000h	45.8.8/1790
24	SDMA Control 1 Register (SPT_SDMA_CTRL1)	32	R/W	0003_0000h	45.8.9/1791
44	Acquisition Status Register A0 (SPT_ACQ_STATUS_A0)	32	R	0000_0000h	45.8.10/ 1792
48	Acquisition Status Register A1 (SPT_ACQ_STATUS_A1)	32	R	8000_7FFCh	45.8.11/ 1793
4C	Acquisition Status Register A2 (SPT_ACQ_STATUS_A2)	32	R	0000_0000h	45.8.12/ 1793
50	Acquisition Status Register B0 (SPT_ACQ_STATUS_B0)	32	R	0000_0000h	45.8.13/ 1794
54	Acquisition Status Register B1 (SPT_ACQ_STATUS_B1)	32	R	8000_7FFCh	45.8.14/ 1794
58	Acquisition Status Register B2 (SPT_ACQ_STATUS_B2)	32	R	0000_0000h	45.8.15/ 1795
5C	Acquisition Status Register C0 (SPT_ACQ_STATUS_C0)	32	R	0000_0000h	45.8.16/ 1796
60	Acquisition Status Register C1 (SPT_ACQ_STATUS_C1)	32	R	8000_7FFCh	45.8.17/ 1796
64	Acquisition Status Register C2 (SPT_ACQ_STATUS_C2)	32	R	0000_0000h	45.8.18/ 1797
68	Acquisition Status Register D0 (SPT_ACQ_STATUS_D0)	32	R	0000_0000h	45.8.19/ 1797

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6C	Acquisition Status Register D1 (SPT_ACQ_STATUS_D1)	32	R	8000_7FFCh	45.8.20/ 1798
70	Acquisition Status Register D2 (SPT_ACQ_STATUS_D2)	32	R	0000_0000h	45.8.21/ 1798
74	Acquisition Status Register E0 (SPT_ACQ_STATUS_E0)	32	R	0000_0000h	45.8.22/ 1799
78	Acquisition Status Register E1 (SPT_ACQ_STATUS_E1)	32	R	8000_7FFCh	45.8.23/ 1799
7C	Acquisition Status Register E2 (SPT_ACQ_STATUS_E2)	32	R	0000_0000h	45.8.24/ 1800
80	Acquisition Status Register F0 (SPT_ACQ_STATUS_F0)	32	R	0000_0000h	45.8.25/ 1800
84	Acquisition Status Register F1 (SPT_ACQ_STATUS_F1)	32	R	8000_7FFCh	45.8.26/ 1801
88	Acquisition Status Register F2 (SPT_ACQ_STATUS_F2)	32	R	0000_0000h	45.8.27/ 1801
8C	Acquisition Status Register G0 (SPT_ACQ_STATUS_G0)	32	R	0000_0000h	45.8.28/ 1802
90	Acquisition Status Register G1 (SPT_ACQ_STATUS_G1)	32	R	8000_7FFCh	45.8.29/ 1802
94	Acquisition Status Register G2 (SPT_ACQ_STATUS_G2)	32	R	0000_0000h	45.8.30/ 1803
98	Acquisition Status Register H0 (SPT_ACQ_STATUS_H0)	32	R	0000_0000h	45.8.31/ 1803
9C	Acquisition Status Register H1 (SPT_ACQ_STATUS_H1)	32	R	8000_7FFCh	45.8.32/ 1804
A0	Acquisition Status Register H2 (SPT_ACQ_STATUS_H2)	32	R	0000_0000h	45.8.33/ 1804
A4	Acquisition MIPICSI2 control register (SPT_ACQ_CSI_CTRL)	32	R/W	0000_0000h	45.8.34/ 1805
A8	Bypass SDMA Control 0 (SPT_SDMA_BYP_CTRL0)	32	R/W	0000_0000h	45.8.35/ 1806
AC	Bypass SDMA Control 1 Register (SPT_SDMA_BYP_CTRL1)	32	R/W	0003_0000h	45.8.36/ 1807
B0	Acquisition Bypass Control Register (SPT_ACQ_BYP_CTRL)	32	R/W	001F_0000h	45.8.37/ 1807
C0	Program Start Addr Register (SPT_CS_PG_ST_ADDR)	32	R/W	0000_0000h	45.8.38/ 1808
C4	Mode Control Register (SPT_CS_MODE_CTRL)	32	R/W	0005_8000h	45.8.39/ 1809
C8	Watchdog Counter Register (SPT_CS_WD_COUNT)	32	R	0000_0000h	45.8.40/ 1811
CC	Breakpoint0 Addr Register (SPT_CS_BKPT0_ADDR)	32	R/W	0000_0000h	45.8.41/ 1811

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D0	Breakpoint1 Addr Register (SPT_CS_BKPT1_ADDR)	32	R/W	0000_0000h	45.8.42/ 1812
D4	Breakpoint2 Addr Register (SPT_CS_BKPT2_ADDR)	32	R/W	0000_0000h	45.8.43/ 1812
D8	Breakpoint3 Addr Register (SPT_CS_BKPT3_ADDR)	32	R/W	0000_0000h	45.8.44/ 1813
DC	Jamming Instruction0 Register (SPT_CS_JAM_INST0)	32	R/W	0000_0000h	45.8.45/ 1813
E0	Jamming Instruction1 Register (SPT_CS_JAM_INST1)	32	R/W	0000_0000h	45.8.46/ 1813
E4	Jamming Instruction2 Register (SPT_CS_JAM_INST2)	32	R/W	0000_0000h	45.8.47/ 1814
E8	Jamming Instruction3 Register (SPT_CS_JAM_INST3)	32	R/W	0000_0000h	45.8.48/ 1814
EC	Current Instruction Addr Register (SPT_CS_CURR_INST_ADDR)	32	R	0000_0000h	45.8.49/ 1815
F0	Current Instruction0 Register (SPT_CS_CURR_INST0)	32	R	0000_0000h	45.8.50/ 1815
F4	Current Instruction1 Register (SPT_CS_CURR_INST1)	32	R	0000_0000h	45.8.51/ 1816
F8	Current Instruction2 Register (SPT_CS_CURR_INST2)	32	R	0000_0000h	45.8.52/ 1816
FC	Current Instruction3 Register (SPT_CS_CURR_INST3)	32	R	0000_0000h	45.8.53/ 1816
100	Loop Counter 0 and 1 Register (SPT_CS_LOOPCNTR01)	32	R	0000_0000h	45.8.54/ 1817
104	Loop Counter 2 and 3 Register (SPT_CS_LOOPCNTR23)	32	R	0000_0000h	45.8.55/ 1817
108	Error Instruction Addr Register (SPT_CS_ERR_INST_ADDR)	32	R	0000_0000h	45.8.56/ 1818
10C	Error Instruction 0 Register (SPT_CS_ERR_INST0)	32	R	0000_0000h	45.8.57/ 1818
110	Error Instruction 1 Register (SPT_CS_ERR_INST1)	32	R	0000_0000h	45.8.58/ 1819
114	Error Instruction 2 Register (SPT_CS_ERR_INST2)	32	R	0000_0000h	45.8.59/ 1819
118	Error Instruction 3 Register (SPT_CS_ERR_INST3)	32	R	0000_0000h	45.8.60/ 1819
11C	General Status 0 Register (SPT_CS_STATUS0)	32	R	0000_1C00h	45.8.61/ 1820
120	General Status 1 Register (SPT_CS_STATUS1)	32	R	0000_0000h	45.8.62/ 1823
124	General Status2 Register (SPT_CS_STATUS2)	32	w1c	0000_0000h	45.8.63/ 1825

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
128	General Status 3 Register (SPT_CS_STATUS3)	32	R	0000_0000h	45.8.64/ 1828
12C	EVT1 Status Register (SPT_CS_EVTREG1)	32	w1c	0000_0000h	45.8.65/ 1829
130	EVT2 Status Register (SPT_CS_EVTREG2)	32	w1c	0000_0000h	45.8.66/ 1830
134	SW Event Trigger Register (SPT_CS_SW_EVTREG)	32	R/W	0000_0000h	45.8.67/ 1830
138	Core 1 Version Register Events (SPT_CORE1_VER_EVT)	32	w1c	0000_0000h	45.8.68/ 1831
13C	Core 2 Version Register Events (SPT_CORE2_VER_EVT)	32	w1c	0000_0000h	45.8.69/ 1833
140	SPT Event Reset Control Register (SPT_EVENT_RST_CTRL)	32	w1c	0000_0000h	45.8.70/ 1835
180	LFSR Load High Value (SPT_PDMA_LFSR_LOAD_VAL_HIGH)	32	R/W	0000_0000h	45.8.71/ 1836
184	LFSR Load Low Value (SPT_PDMA_LFSR_LOAD_VAL_LOW)	32	R/W	0000_0000h	45.8.72/ 1836
188	PDMA Control Register (SPT_PDMA_CONTROL)	32	R/W	FFFF_0008h	45.8.73/ 1837
1C0	PDMA Transfer Count Status (SPT_PDMA_TRANSFER_COUNT_STATUS)	32	R/W	0000_0000h	45.8.74/ 1838
1C4	PDMA formatB Exponent Address status Register (SPT_PDMA_FMTB_EXP_ADDR_STATUS)	32	R	0000_0000h	45.8.75/ 1838
1FC	Memory Error Injection Register (SPT_MEM_ERR_INJECT_CTRL)	32	w1c	0000_0000h	45.8.76/ 1839
200	Memory Error Status Register (SPT_MEM_ERR_STATUS)	32	w1c	0000_0000h	45.8.77/ 1840
204	Memory Interrupt Enable register (SPT_MEM_ERR_INT_EN)	32	R/W	0000_0000h	45.8.78/ 1841
208	DMA Error Status Register (SPT_DMA_ERR_STATUS)	32	w1c	0000_0000h	45.8.79/ 1843
20C	Interrupt Enable for DMA_ERROR_STATUS (SPT_DMA_ERR_INT_EN)	32	R/W	0000_0000h	45.8.80/ 1845
210	Global Status Register (SPT_GBL_STATUS)	32	w1c	0000_0000h	45.8.81/ 1847
214	Global Status Interrupt Enable Register (SPT_GBL_STATUS_IE)	32	R/W	0000_0000h	45.8.82/ 1848
218	Hardware Accelerator Error Status Register (SPT_HW_ACC_ERR_STATUS)	32	w1c	0000_0000h	45.8.83/ 1850
21C	Hardware Accelerator Error Interrupt Enable Register (SPT_HW_ACC_ERR_IE)	32	R/W	0000_0000h	45.8.84/ 1854
220	HIST Overflow Status0 Register (SPT_HIST_OVF_STATUS0)	32	w1c	0000_0000h	45.8.85/ 1857

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
224	HIST Overflow Status1 Register (SPT_HIST_OVF_STATUS1)	32	w1c	0000_0000h	45.8.86/ 1859
228	HIST Overflow Interrupt Enable Register (SPT_HIST_OVF_IE)	32	R/W	0000_0000h	45.8.87/ 1862
22C	Interrupt Enable Register 0 (SPT_CS_INTEN0)	32	R/W	0000_0000h	45.8.88/ 1863
230	Interrupt Enable Register 1 (SPT_CS_INTEN1)	32	R/W	0000_0000h	45.8.89/ 1865
234	EVT1 Interrupt Enable Register (SPT_CS_EVT1_INTEN)	32	R/W	0000_0000h	45.8.90/ 1866
238	EVT2 Interrupt Enable Register (SPT_CS_EVT2_INTEN)	32	R/W	0000_0000h	45.8.91/ 1866
240	SPT_WR_0_15_CTRL_REG	32	R/W	0000_0000h	45.8.92/ 1867
244	SPT_WR_16_31_CTRL_REG	32	R/W	0000_0000h	45.8.93/ 1871
248	SPT_WR_32_47_CTRL_REG	32	R/W	0000_0000h	45.8.94/ 1875
250	Work Register R0 Real (SPT_WR_R0_RE)	32	R/W	0000_0000h	45.8.95/ 1879
254	Work Register R0 Imaginary (SPT_WR_R0_IM)	32	R/W	0000_0000h	45.8.96/ 1879
258	Work Register R1 Real (SPT_WR_R1_RE)	32	R/W	0000_0000h	45.8.97/ 1880
25C	Work Register R1 Imaginary (SPT_WR_R1_IM)	32	R/W	0000_0000h	45.8.98/ 1880
260	Work Register R0 Real (SPT_WR_R2_RE)	32	R/W	0000_0000h	45.8.99/ 1881
264	Work Register R2 Imaginary (SPT_WR_R2_IM)	32	R/W	0000_0000h	45.8.100/ 1881
268	Work Register R3 Real (SPT_WR_R3_RE)	32	R/W	0000_0000h	45.8.101/ 1882
26C	Work Register R3 Imaginary (SPT_WR_R3_IM)	32	R/W	0000_0000h	45.8.102/ 1882
270	Work Register R4 Real (SPT_WR_R4_RE)	32	R/W	0000_0000h	45.8.103/ 1883
274	Work Register R4 Imaginary (SPT_WR_R4_IM)	32	R/W	0000_0000h	45.8.104/ 1883
278	Work Register R5 Real (SPT_WR_R5_RE)	32	R/W	0000_0000h	45.8.105/ 1884
27C	Work Register R5 Imaginary (SPT_WR_R5_IM)	32	R/W	0000_0000h	45.8.106/ 1884
280	Work Register R6 Real (SPT_WR_R6_RE)	32	R/W	0000_0000h	45.8.107/ 1885

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
284	Work Register R6 Imaginary (SPT_WR_R6_IM)	32	R/W	0000_0000h	45.8.108/ 1885
288	Work Register R7 Real (SPT_WR_R7_RE)	32	R/W	0000_0000h	45.8.109/ 1886
28C	Work Register R7 Imaginary (SPT_WR_R7_IM)	32	R/W	0000_0000h	45.8.110/ 1886
290	Work Register R8 Real (SPT_WR_R8_RE)	32	R/W	0000_0000h	45.8.111/ 1887
294	Work Register R8 Imaginary (SPT_WR_R8_IM)	32	R/W	0000_0000h	45.8.112/ 1887
298	Work Register R9 Real (SPT_WR_R9_RE)	32	R/W	0000_0000h	45.8.113/ 1888
29C	Work Register R9 Imaginary (SPT_WR_R9_IM)	32	R/W	0000_0000h	45.8.114/ 1888
2A0	Work Register R10 Real (SPT_WR_R10_RE)	32	R/W	0000_0000h	45.8.115/ 1889
2A4	Work Register R10 Imaginary (SPT_WR_R10_IM)	32	R/W	0000_0000h	45.8.116/ 1889
2A8	Work Register R11 Real (SPT_WR_R11_RE)	32	R/W	0000_0000h	45.8.117/ 1890
2AC	Work Register R11 Imaginary (SPT_WR_R11_IM)	32	R/W	0000_0000h	45.8.118/ 1890
2B0	Work Register R12 Real (SPT_WR_R12_RE)	32	R/W	0000_0000h	45.8.119/ 1891
2B4	Work Register R12 Imaginary (SPT_WR_R12_IM)	32	R/W	0000_0000h	45.8.120/ 1891
2B8	Work Register R13 Real (SPT_WR_R13_RE)	32	R/W	0000_0000h	45.8.121/ 1892
2BC	Work Register R13 Imaginary (SPT_WR_R13_IM)	32	R/W	0000_0000h	45.8.122/ 1892
2C0	Work Register R14 Real (SPT_WR_R14_RE)	32	R/W	0000_0000h	45.8.123/ 1893
2C4	Work Register R14 Imaginary (SPT_WR_R14_IM)	32	R/W	0000_0000h	45.8.124/ 1893
2C8	Work Register R15 Real (SPT_WR_R15_RE)	32	R/W	0000_0000h	45.8.125/ 1894
2CC	Work Register R15 Imaginary (SPT_WR_R15_IM)	32	R/W	0000_0000h	45.8.126/ 1894
2D0	Work Register R16 Real (SPT_WR_R16_RE)	32	R/W	0000_0000h	45.8.127/ 1895
2D4	Work Register R16 Imaginary (SPT_WR_R16_IM)	32	R/W	0000_0000h	45.8.128/ 1895
2D8	Work Register R17 Real (SPT_WR_R17_RE)	32	R/W	0000_0000h	45.8.129/ 1896

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2DC	Work Register R17 Imaginary (SPT_WR_R17_IM)	32	R/W	0000_0000h	45.8.130/ 1896
2E0	Work Register R18 Real (SPT_WR_R18_RE)	32	R/W	0000_0000h	45.8.131/ 1897
2E4	Work Register R18 Imaginary (SPT_WR_R18_IM)	32	R/W	0000_0000h	45.8.132/ 1897
2E8	Work Register R19 Real (SPT_WR_R19_RE)	32	R/W	0000_0000h	45.8.133/ 1898
2EC	Work Register R19 Imaginary (SPT_WR_R19_IM)	32	R/W	0000_0000h	45.8.134/ 1898
2F0	Work Register R20 Real (SPT_WR_R20_RE)	32	R/W	0000_0000h	45.8.135/ 1899
2F4	Work Register R20 Imaginary (SPT_WR_R20_IM)	32	R/W	0000_0000h	45.8.136/ 1899
2F8	Work Register R21 Real (SPT_WR_R21_RE)	32	R/W	0000_0000h	45.8.137/ 1900
2FC	Work Register R21 Imaginary (SPT_WR_R21_IM)	32	R/W	0000_0000h	45.8.138/ 1900
300	Work Register R22 Real (SPT_WR_R22_RE)	32	R/W	0000_0000h	45.8.139/ 1901
304	Work Register R22 Imaginary (SPT_WR_R22_IM)	32	R/W	0000_0000h	45.8.140/ 1901
308	Work Register R23 Real (SPT_WR_R23_RE)	32	R/W	0000_0000h	45.8.141/ 1902
30C	Work Register R23 Imaginary (SPT_WR_R23_IM)	32	R/W	0000_0000h	45.8.142/ 1902
310	Work Register R24 Real (SPT_WR_R24_RE)	32	R/W	0000_0000h	45.8.143/ 1903
314	Work Register R24 Imaginary (SPT_WR_R24_IM)	32	R/W	0000_0000h	45.8.144/ 1903
318	Work Register R25 Real (SPT_WR_R25_RE)	32	R/W	0000_0000h	45.8.145/ 1904
31C	Work Register R25 Imaginary (SPT_WR_R25_IM)	32	R/W	0000_0000h	45.8.146/ 1904
320	Work Register R26 Real (SPT_WR_R26_RE)	32	R/W	0000_0000h	45.8.147/ 1905
324	Work Register R26 Imaginary (SPT_WR_R26_IM)	32	R/W	0000_0000h	45.8.148/ 1905
328	Work Register R27 Real (SPT_WR_R27_RE)	32	R/W	0000_0000h	45.8.149/ 1906
32C	Work Register R27 Imaginary (SPT_WR_R27_IM)	32	R/W	0000_0000h	45.8.150/ 1906
330	Work Register R28 Real (SPT_WR_R28_RE)	32	R/W	0000_0000h	45.8.151/ 1907

Table continues on the next page...

SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
334	Work Register R28 Imaginary (SPT_WR_R28_IM)	32	R/W	0000_0000h	45.8.152/ 1907
338	Work Register R29 Real (SPT_WR_R29_RE)	32	R/W	0000_0000h	45.8.153/ 1908
33C	Work Register R29 Imaginary (SPT_WR_R29_IM)	32	R/W	0000_0000h	45.8.154/ 1908
340	Work Register R30 Real (SPT_WR_R30_RE)	32	R/W	0000_0000h	45.8.155/ 1909
344	Work Register R30 Imaginary (SPT_WR_R30_IM)	32	R/W	0000_0000h	45.8.156/ 1909
348	Work Register R31 Real (SPT_WR_R31_RE)	32	R/W	0000_0000h	45.8.157/ 1910
34C	Work Register R31 Imaginary (SPT_WR_R31_IM)	32	R/W	0000_0000h	45.8.158/ 1910
350	Work Register R32 Real (SPT_WR_R32_RE)	32	R/W	0000_0000h	45.8.159/ 1911
354	Work Register R32 Imaginary (SPT_WR_R32_IM)	32	R/W	0000_0000h	45.8.160/ 1911
358	Work Register R33 Real (SPT_WR_R33_RE)	32	R/W	0000_0000h	45.8.161/ 1912
35C	Work Register R33 Imaginary (SPT_WR_R33_IM)	32	R/W	0000_0000h	45.8.162/ 1912
360	Work Register R34 Real (SPT_WR_R34_RE)	32	R/W	0000_0000h	45.8.163/ 1913
364	Work Register R34 Imaginary (SPT_WR_R34_IM)	32	R/W	0000_0000h	45.8.164/ 1913
368	Work Register R35 Real (SPT_WR_R35_RE)	32	R/W	0000_0000h	45.8.165/ 1914
36C	Work Register R35 Imaginary (SPT_WR_R35_IM)	32	R/W	0000_0000h	45.8.166/ 1914
370	Work Register R36 Real (SPT_WR_R36_RE)	32	R/W	0000_0000h	45.8.167/ 1915
374	Work Register R36 Imaginary (SPT_WR_R36_IM)	32	R/W	0000_0000h	45.8.168/ 1915
378	Work Register R37 Real (SPT_WR_R37_RE)	32	R/W	0000_0000h	45.8.169/ 1916
37C	Work Register R37 Imaginary (SPT_WR_R37_IM)	32	R/W	0000_0000h	45.8.170/ 1916
380	Work Register R38 Real (SPT_WR_R38_RE)	32	R/W	0000_0000h	45.8.171/ 1917
384	Work Register R38 Imaginary (SPT_WR_R38_IM)	32	R/W	0000_0000h	45.8.172/ 1917
388	Work Register R39 Real (SPT_WR_R39_RE)	32	R/W	0000_0000h	45.8.173/ 1918

Table continues on the next page...

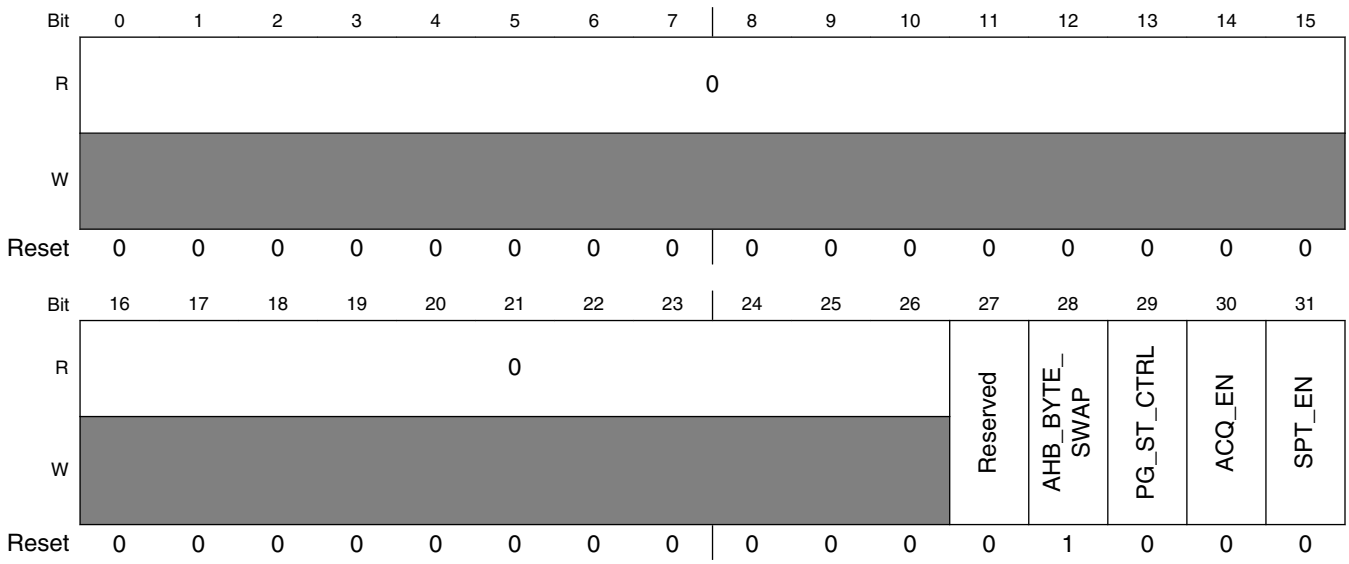
SPT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38C	Work Register R39 Imaginary (SPT_WR_R39_IM)	32	R/W	0000_0000h	45.8.174/ 1918
390	Work Register R40 Real (SPT_WR_R40_RE)	32	R/W	0000_0000h	45.8.175/ 1919
394	Work Register R40 Imaginary (SPT_WR_R40_IM)	32	R/W	0000_0000h	45.8.176/ 1919
398	Work Register R41 Real (SPT_WR_R41_RE)	32	R/W	0000_0000h	45.8.177/ 1920
39C	Work Register R41 Imaginary (SPT_WR_R41_IM)	32	R/W	0000_0000h	45.8.178/ 1920
3A0	Work Register R42 Real (SPT_WR_R42_RE)	32	R/W	0000_0000h	45.8.179/ 1921
3A4	Work Register R42 Imaginary (SPT_WR_R42_IM)	32	R/W	0000_0000h	45.8.180/ 1921
3A8	Work Register R43 Real (SPT_WR_R43_RE)	32	R/W	0000_0000h	45.8.181/ 1922
3AC	Work Register R43 Imaginary (SPT_WR_R43_IM)	32	R/W	0000_0000h	45.8.182/ 1922
3B0	Work Register R44 Real (SPT_WR_R44_RE)	32	R/W	0000_0000h	45.8.183/ 1923
3B4	Work Register R44 Imaginary (SPT_WR_R44_IM)	32	R/W	0000_0000h	45.8.184/ 1923
3B8	Work Register R45 Real (SPT_WR_R45_RE)	32	R/W	0000_0000h	45.8.185/ 1924
3BC	Work Register R45 Imaginary (SPT_WR_R45_IM)	32	R/W	0000_0000h	45.8.186/ 1924
3C0	Work Register R46 Real (SPT_WR_R46_RE)	32	R/W	0000_0000h	45.8.187/ 1925
3C4	Work Register R46 Imaginary (SPT_WR_R46_IM)	32	R/W	0000_0000h	45.8.188/ 1925
3C8	Work Register R47 Real (SPT_WR_R47_RE)	32	R/W	0000_0000h	45.8.189/ 1926
3CC	Work Register R47 Imaginary (SPT_WR_R47_IM)	32	R/W	0000_0000h	45.8.190/ 1926

45.8.1 SPT Global Control Register (SPT_GBL_CTRL)

This register allows user to configure various control fields as described below.

Address: 0h base + 0h offset = 0h



SPT_GBL_CTRL field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.
28 AHB_BYTE_ SWAP	Bit to control data byte swapping on the AHB interface. 1 - Swap(required). This bit is Reserved and always write as 1.
29 PG_ST_CTRL	Program start control Program execution starts only after this bit is set. When this bit is set the command sequencer in the START state checks if there is any residual instruction fetch operation still going on. If no such operation is going on then the sequencer moves to the SETUP state and this bit is internally cleared. This bit should be set when SPT_EN is high. When set, user should wait for PG_ST_CTRL to be 0 to ensure that all status are cleared properly.
30 ACQ_EN	Acquisition Enable. This field enables the Acquisition module. This field must be asserted only after all other Acquisition and SDMA register fields have been programmed. This bit should be set after IPS programming is done for the SPT domain modules. When this bit is cleared, it will reset all the ACQ and SDMA modules after any on-going AHB burst is completed. ACQ_EN and SPT_EN bit can be used independently. Before changing the configuration for Acquisition ACQ_EN should be deasserted. Note: There must be a gap of atleast 10 AFE/MIPICSI2 interface clock cycles between setting this bit and subsequently asserting of frame start signal. There must be a gap of atleast 100 AFE/MIPICSI2 interface clock cycles between resetting this bit and subsequently setting it; so that there is enough time for any on-going AHB burst to complete and for ACQ and SDMA modules to be reset.

Table continues on the next page...

SPT_GBL_CTRL field descriptions (continued)

Field	Description
	0 Acquisition module is disabled. Acquisition and SDMA registers can now be programmed. 1 Acquisition module is enabled. ADC samples can now be collected by acquisition block which are processed and transferred to System RAM by SDMA via AHB bus.
31 SPT_EN	SPT Enable This bit should be set after IPS programming is done for the SPT domain modules. When reset, any on-going CSDMA or PDMA burst is completed and thereafter it will reset all the SPT domain modules (which is all modules except ACQ, SDMA and AHB Arbiter). This bit can be used independently from ACQ_EN. Note: There must be a gap of atleast 100 AHB interface clock cycles between resetting this bit and setting it again; so that there is enough time for any on-going AHB burst to complete and for SPT domain modules to be reset. Also, please note that this bit should be reset only when the command sequencer FSM is in the start state (One way to ensure this is by setting the SPT_CS_MODE_CTRL[CFG_STOP] or SPT_CS_MODE_CTRL[ASYNCSTOP] bit and then poll the SPT_CS_STATUS3[PROC_STATE] field). 0 SPT is disabled 1 SPT is enabled.

45.8.2 Acquisition Global Control Register (SPT_ACQ_GBL_CTRL_0)

This register allows user to configure various control fields as described below.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH_H_SEL		CH_G_SEL		CH_F_SEL		CH_E_SEL		CH_D_SEL		CH_C_SEL		CH_B_SEL		CH_A_SEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSI_ADC_SEL	INF_CHRPS	INF_SMPLS	ADC_SWAP_SEL	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_GBL_CTRL_0 field descriptions

Field	Description
0–1 CH_H_SEL	Channel H Select This field is used to disable or swap channel H. If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. This field should be set to zero if its real data mode or 4ADC mode.

Table continues on the next page...

SPT_ACQ_GBL_CTRL_0 field descriptions (continued)

Field	Description
	00 Channel Disabled 01 No Swap - Channel H 10 Swaps to Channel A 11 Swaps to Channel G
2–3 CH_G_SEL	Channel G Select This field is used to disable or swap channel G. If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped.This field should be set to zero if its real data mode or 4ADC mode. 00 Channel Disabled 01 No Swap - Channel G 10 Swaps to Channel B 11 Swaps to Channel H
4–5 CH_F_SEL	Channel F Select This field is used to disable or swap channel F. If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. This field should be set to zero if its real data mode or 4ADC mode. 00 Channel Disabled 01 No Swap - Channel F 10 Swaps to Channel C 11 Swaps to Channel E
6–7 CH_E_SEL	Channel E Select This field is used to disable or swap channel E. If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. This field should be set to zero if its real data mode or 4ADC mode. 00 Channel Disabled 01 No Swap - Channel E 10 Swaps to Channel D 11 Swaps to Channel F
8–9 CH_D_SEL	Channel D Select This field is used to disable or swap channel D for ADC data. The channel swapping varies from 4 to 8 ADC channels, given by ADC_SWAP_SEL If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. 00 Channel Disabled 01 No Swap - Channel D 10 Swaps to Channel C 11 Swaps to Channel E for 8ADCs, swaps to Channel A for 4ADCs
10–11 CH_C_SEL	Channel C Select This field is used to disable or swap channel C for ADC data. The channel swapping varies from 4 to 8 ADC channels, given by ADC_SWAP_SEL If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped.

Table continues on the next page...

SPT_ACQ_GBL_CTRL_0 field descriptions (continued)

Field	Description
	00 Channel Disabled 01 No Swap - Channel C 10 Swaps to Channel DReserved 11 Swaps to Channel F for 8ADCs, swaps to channel B for 4ADCs
12–13 CH_B_SEL	Channel B Select This field is used to disable or swap channel B for ADC data.The channel swapping varies from 4 to 8 ADC channels, given by ADC_SWAP_SEL If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. 00 Channel Disabled 01 No Swap - Channel B 10 Swaps to Channel A 11 Swaps to Channel G for 8ADCs, swaps to channel C for 4ADCs
14–15 CH_A_SEL	Channel A Select This field can be used to disable or swap channel A for ADC data.The channel swapping varies from 4 to 8 ADC channels, given by ADC_SWAP_SEL If the MIPICSI2 path is enabled, the channel can only be disabled and not swapped. 00 Channel Disabled 01 No Swap - Channel A 10 Swaps to Channel B 11 Swaps to Channel H for 8ADCs, swaps to channel D for 4ADCs
16 CSI_ADC_SEL	CSI/ADC Select This field decides between using ADC data or MIPICSI2 data as source for Acquisition. 0 Acquisition source is ADC data 1 acquisition source is MIPICSI2 data
17 INF_CHRPS	Infinite Chirps in a Frame This field allows user to override ACQ_GBL_CTRL_1[NUM_CHRP] value and allows infinite chirps in a frame. Asserting this field prevents end of frame from being detected, thus preventing GBL_STA[FRM_ACQ_DONE] from asserting. User must wait for GBL_STA[CHRP_ACQ_DONE] after last chirp in the frame has been processed, before performing a read on any statistics present on Acquisition Status registers. 0 The number of chirps within a frame is limited and is programmable via ACQ_GBL_CTRL_1[NUM_CHRP] field. 1 The number of chirps within a frame is unlimited.
18 INF_SMPLS	Infinite Samples in a Chirp This field allows user to override ACQ_GBL_CTRL_1[NUM_SMPL] value and allows infinite samples in a chirp. If asserted, the acquisition window signal from CTE/MIPICSI2 solely decides the window for acquisition within a chirp. It is however expected that the acquisition window will be deasserted only after equal samples of all channels have been acquired. 0 The number of samples within a chirp is limited and is programmable via ACQ_GBL_CTRL_1[NUM_SMPL] field. 1 The number of samples within a chirp is unlimited.

Table continues on the next page...

SPT_ACQ_GBL_CTRL_0 field descriptions (continued)

Field	Description
19 ADC_SWAP_SEL	ADC Swap select Channel Swap select for 4 or 8 ADCs. **NOTE:** Recommended setting of ADC_SWAP_SEL is 1, as SDADC has 4 channels. 0 8 ADC channel swapping 1 4 ADC channel swapping
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.3 Acquisition Global Control Register 1 (SPT_ACQ_GBL_CTRL_1)

This register allows user to configure various control fields as described below.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NUM_SMPL																NUM_CHRP															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

SPT_ACQ_GBL_CTRL_1 field descriptions

Field	Description
0–15 NUM_SMPL	Number of Samples in a Chirp This value is given by (NUM_SMPL + 1)*8 for each channel, hence configuring this field to 0 results in 8 samples per chirp. Similarly the default value of 31 corresponds to 256 samples per chirp. Maximum value of 65,535 corresponds to 524,288 samples per chirp. This field is invalid if ACQ_GBL_CTRL_0[INF_SMPLS] is asserted.
16–31 NUM_CHRP	Number of chirps in a frame The number of chirps in a frame is given directly by this field Configuring this field to 0 results in 0 chirps per frame. Similarly the default value of 31 corresponds to 31 chirps per frame. This field can be configured upto a maximum of 65,535 chirps per frame. The chirp count remains the same for both SDMAs. This field is invalid if ACQ_GBL_CTRL_0[INF_CHRPS] is asserted.

45.8.4 Acquisition Control 0 Register (SPT_ACQ_CTRL0)

Gives offset compensation values for channels A and B.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CH_A_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	CH_B_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_CTRL0 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CH_A_DC	Channel A Offset Compensation value This is a signed 15-bit value
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 CH_B_DC	Channel B Offset Compensation value This is a signed 15-bit value

45.8.5 Acquisition Control 1 Register (SPT_ACQ_CTRL1)

Gives offset compensation values for channels C and D.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CH_C_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	CH_D_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_CTRL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CH_C_DC	Channel C Offset Compensation value This is a signed 15-bit value
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 CH_D_DC	Channel D Offset Compensation value This is a signed 15-bit value

45.8.6 Acquisition Control 2 Register (SPT_ACQ_CTRL2)

Gives offset compensation values for channels E and F.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CH_E_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	CH_F_DC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

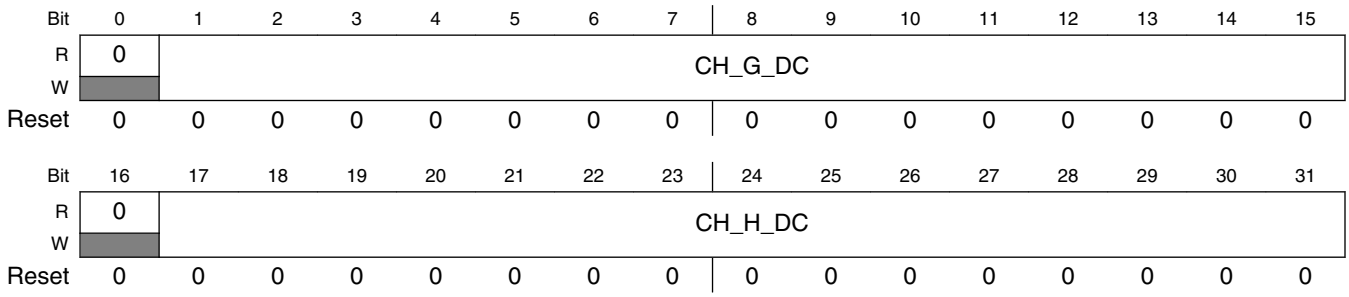
SPT_ACQ_CTRL2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CH_E_DC	Channel E Offset Compensation value This is a signed 15-bit value
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 CH_F_DC	Channel F Offset Compensation value This is a signed 15-bit value

45.8.7 Acquisition Control 3 Register (SPT_ACQ_CTRL3)

Gives offset Compensation values for channels G and H.

Address: 0h base + 1Ch offset = 1Ch

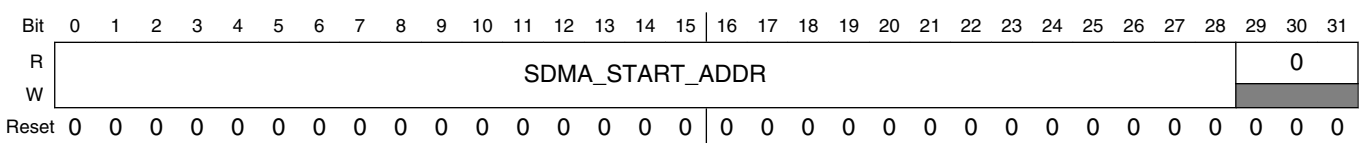


SPT_ACQ_CTRL3 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CH_G_DC	Channel G Offset Compensation value This is a signed 15-bit value
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 CH_H_DC	Channel H Offset Compensation value This is a signed 15-bit value

45.8.8 SDMA Control 0 Register (SPT_SDMA_CTRL0)

Address: 0h base + 20h offset = 20h

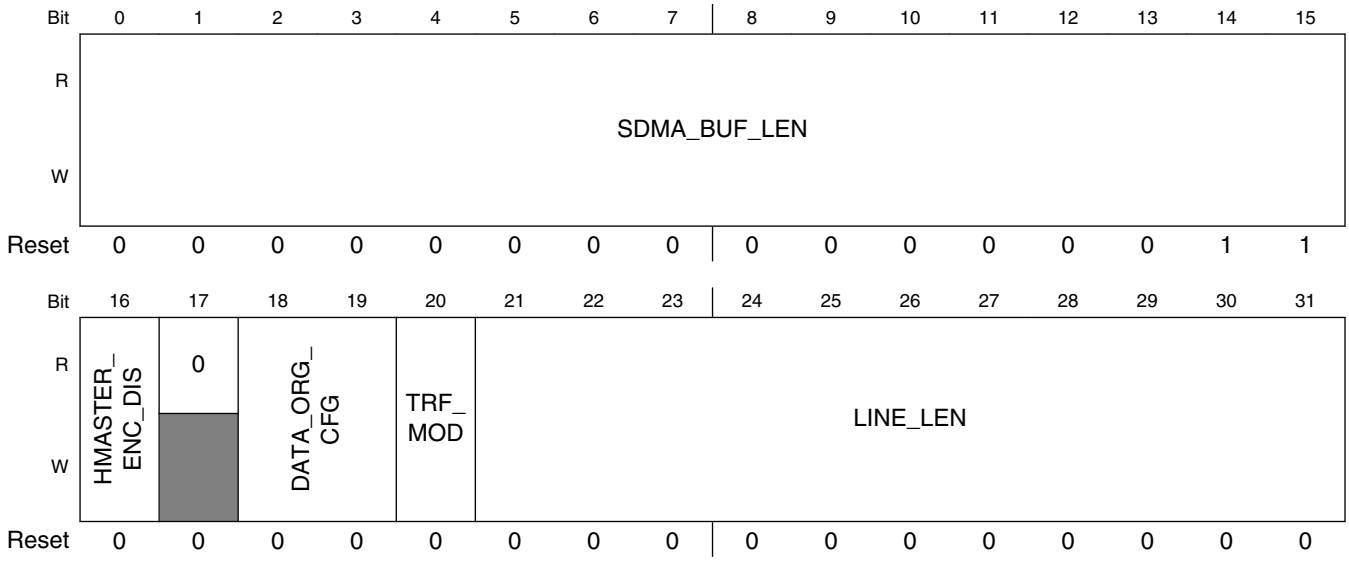


SPT_SDMA_CTRL0 field descriptions

Field	Description
0–28 SDMA_START_ADDR	SDMA Start Address All 32 bits in this register together provide the System RAM address from which SDMA starts writing ADC packets. The 3 LSB's of this address are restricted to zero, since this address must be a multiple of 8.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.9 SDMA Control 1 Register (SPT_SDMA_CTRL1)

Address: 0h base + 24h offset = 24h



SPT_SDMA_CTRL1 field descriptions

Field	Description
0–15 SDMA_BUF_ LEN	SDMA Buffer Length This field provides the number of chirps in which samples received can be stored in System RAM continuously without being over-written. Once this is done, wrap-around occurs and previous samples in System RAM will be over-written. The default value of this field is 3. A value of 0 is invalid and hence should not be programmed in this field. For example, if this field is set to 2, then samples received from first 2 chirps are stored in SRAM one after the other depending on the active SDMA addressing mode. The SDMA write address resets to SPT_SDMA_CTRL0[SDMA_START_ADDRESS] from 3'rd chirp onwards, over-writing samples from previous two chirps with samples from 3'rd and 4'th chirps respectively.
16 HMASTER_ ENC_DIS	AHB Master ID generation disable Note: If SPT_SDMA_CTRL1[HMASTER_ENC_DIS] bit is 1, then the Duplicate Master ID will be 0xE when regular SDMA is giving out its burst and will 0xA when the Bypass SDMA is giving out its data. If SPT_SDMA_CTRL1[HMASTER_ENC_DIS] bit is 0, then whenever SDMA transfer is ongoing the Duplicate Master ID can be used to decode which ADC or MIPICSI2 source's data is being transferred in the transaction. If Interleaved mode is selected, the Duplicate Master ID will be: • 0x0 => First 4 channel's data in the transaction. • 0x1 => Last 4 channel's data in the transaction. If Tile-4 or Tile-8 mode is selected, the MasterID will be: • 0x0 => 1st enabled channel. • 0x1 => 2nd enabled channel. • 0x2 => 3rd enabled channel. • 0x3 => 4th enabled channel. • 0x8 => 5th enabled channel. • 0x9 => 6th enabled channel.

Table continues on the next page...

SPT_SDMA_CTRL1 field descriptions (continued)

Field	Description
	• 0xA => 7th enabled channel. • 0xB => 8th enabled channel. For MIPICSI2 real data, first 4 channels are used and for MIPICSI2 complex data all 8 channels are used. For bypass SDMA, the duplicate Master ID is 0xF. Note that the Crossbar Master ID is not affected by SPT_SDMA_CTRL1[HMASTER_ENC_DIS]; only the Duplicate Master ID is affected. The Duplicate Master ID encoding is only for the debugger to receive via NXMC. The PDMA and CSDMA master IDs are not affected by this bit. They will be 0x6 and 0xD respectively. 0 Duplicate AHB Master ID generated. In this case the Duplicate AHB Master ID at the top shows a value different from the Crossbar Master ID 1 Duplicate AHB Master ID not generated. In this case the Duplicate AHB Master ID at the top shows the same value as the Crossbar Master ID
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 DATA_ORG_CFG	Data Organization Configuration This field selects between various configuration options by which regular SDMA places the ADC data in its internal buffer. The bypass SDMA always places the data in interleaved mode for both 5th channel and image data formats irrespective of this bitfield. 00 Interleaved Mode 01 Tile 4 Mode 10 Tile 8 Mode 11 Reserved
20 TRF_MOD	Transfer Mode This Field is used to select between 1D (Linear) and 2D (Orthogonal) modes for SDMA transfer. 0 Selects Linear (1D) Mode 1 Selects Orthogonal (2D) Mode
21–31 LINE_LEN	Line Length This field is utilized when SDMA_CTRL1[TRF_MOD] = 1; i.e. 2D Mode is selected. The value of line length is given by (LINE_LEN + 1)*8 Hence configuring this field to 0 results in line length of 8. Similarly a value of 31 corresponds to line length of 256. Maximum value of 2047 corresponds to line length of 16384.

45.8.10 Acquisition Status Register A0 (SPT_ACQ_STATUS_A0)

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_A_SUM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_A0 field descriptions

Field	Description
0–31 CH_A_SUM	Channel A Sum This field provides the signed sum of all samples received by channel A within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.11 Acquisition Status Register A1 (SPT_ACQ_STATUS_A1)

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_A_MAX																CH_A_MIN															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SPT_ACQ_STATUS_A1 field descriptions

Field	Description
0–15 CH_A_MAX	Channel A Maximum This field provides the global (signed) maximum of all samples received from channel A within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_A_MIN	Channel A Minimum This field provides the global (signed) minimum of all samples received from channel A within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.12 Acquisition Status Register A2 (SPT_ACQ_STATUS_A2)

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CH_A_BT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_A2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_A_BT	Channel A Bit Toggle In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel A input by scanning all samples received from channel A within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.13 Acquisition Status Register B0 (SPT_ACQ_STATUS_B0)

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_B_SUM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_B0 field descriptions

Field	Description
0–31 CH_B_SUM	Channel B Sum This field provides the signed sum of all samples received from channel B within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.14 Acquisition Status Register B1 (SPT_ACQ_STATUS_B1)

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_B_MAX																CH_B_MIN															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SPT_ACQ_STATUS_B1 field descriptions

Field	Description
0–15 CH_B_MAX	Channel B Maximum This field provides the global (signed) maximum of all samples received from channel B within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_B_MIN	Channel B Minimum This field provides the global (signed) minimum of all samples received from channel B within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.15 Acquisition Status Register B2 (SPT_ACQ_STATUS_B2)

Address: 0h base + 58h offset = 58h

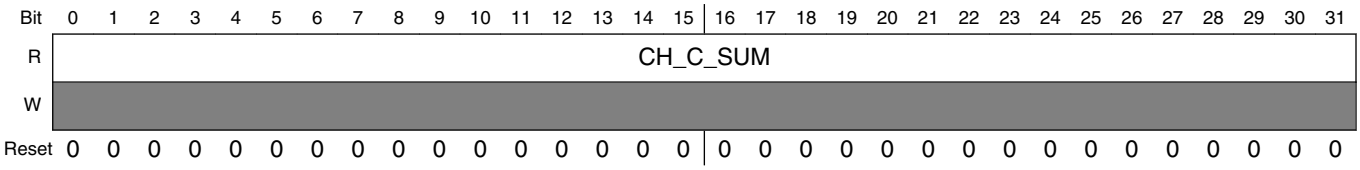
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CH_B_BT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_B2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_B_BT	Channel B Bit Toggle In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel B input by scanning all samples received from channel B within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.16 Acquisition Status Register C0 (SPT_ACQ_STATUS_C0)

Address: 0h base + 5Ch offset = 5Ch

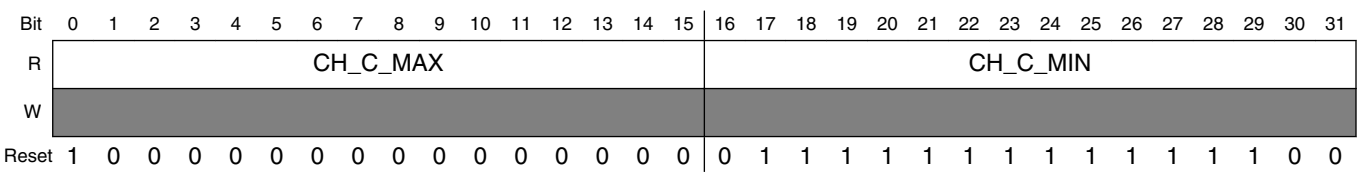


SPT_ACQ_STATUS_C0 field descriptions

Field	Description
0–31 CH_C_SUM	Channel C Sum This field provides the signed sum of all samples received by channel C within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.17 Acquisition Status Register C1 (SPT_ACQ_STATUS_C1)

Address: 0h base + 60h offset = 60h

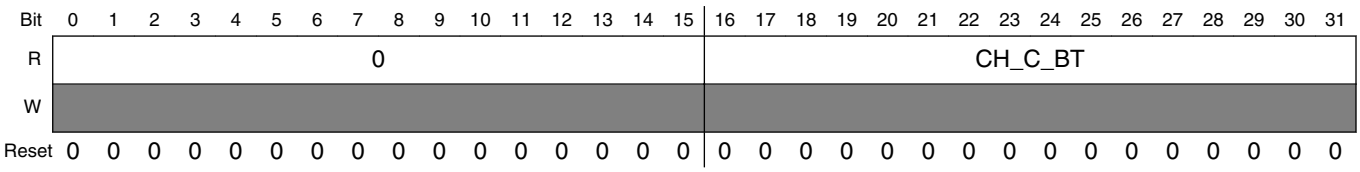


SPT_ACQ_STATUS_C1 field descriptions

Field	Description
0–15 CH_C_MAX	Channel C Maximum This field provides the global (signed) maximum of all samples received from channel C within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_C_MIN	Channel C Minimum This field provides the global (signed) minimum of all samples received from channel C within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.18 Acquisition Status Register C2 (SPT_ACQ_STATUS_C2)

Address: 0h base + 64h offset = 64h

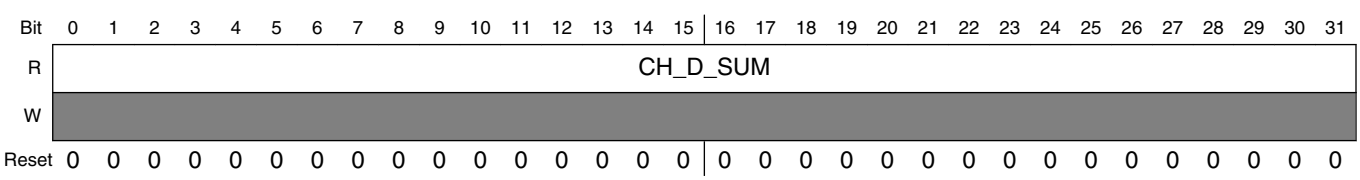


SPT_ACQ_STATUS_C2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_C_BT	Channel C Bit Toggle In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel C input by scanning all samples received from channel C within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.19 Acquisition Status Register D0 (SPT_ACQ_STATUS_D0)

Address: 0h base + 68h offset = 68h



SPT_ACQ_STATUS_D0 field descriptions

Field	Description
0–31 CH_D_SUM	Channel D Sum This field provides the signed sum of all samples received by channel D within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.20 Acquisition Status Register D1 (SPT_ACQ_STATUS_D1)

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_D_MAX																CH_D_MIN															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SPT_ACQ_STATUS_D1 field descriptions

Field	Description
0–15 CH_D_MAX	Channel D Maximum This field provides the global (signed) maximum of all samples received from channel D within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_D_MIN	Channel D Minimum This field provides the global (signed) minimum of all samples received from channel D within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.21 Acquisition Status Register D2 (SPT_ACQ_STATUS_D2)

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CH_D_BT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_D2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_D_BT	Channel D Bit Toggle In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel D input by scanning all samples received from channel D within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros.

Table continues on the next page...

SPT_ACQ_STATUS_D2 field descriptions (continued)

Field	Description
	0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.22 Acquisition Status Register E0 (SPT_ACQ_STATUS_E0)

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_E_SUM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_E0 field descriptions

Field	Description
0–31 CH_E_SUM	Channel E Sum This field provides the signed sum of all samples received by channel E within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.23 Acquisition Status Register E1 (SPT_ACQ_STATUS_E1)

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_E_MAX																CH_E_MIN															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SPT_ACQ_STATUS_E1 field descriptions

Field	Description
0–15 CH_E_MAX	Channel E Maximum This field provides the global (signed) maximum of all samples received from channel E within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_E_MIN	Channel E Minimum This field provides the global (signed) minimum of all samples received from channel E within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between

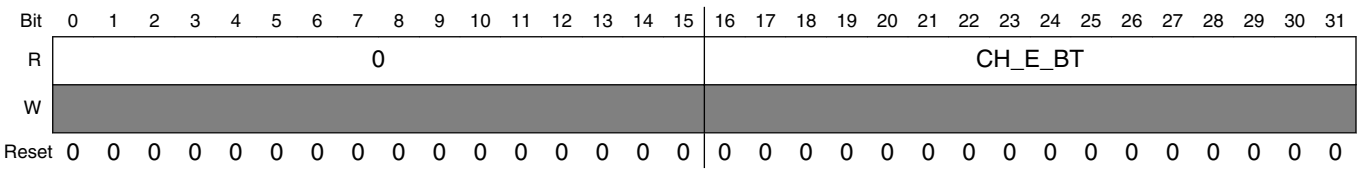
Table continues on the next page...

SPT_ACQ_STATUS_E1 field descriptions (continued)

Field	Description
	assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.24 Acquisition Status Register E2 (SPT_ACQ_STATUS_E2)

Address: 0h base + 7Ch offset = 7Ch

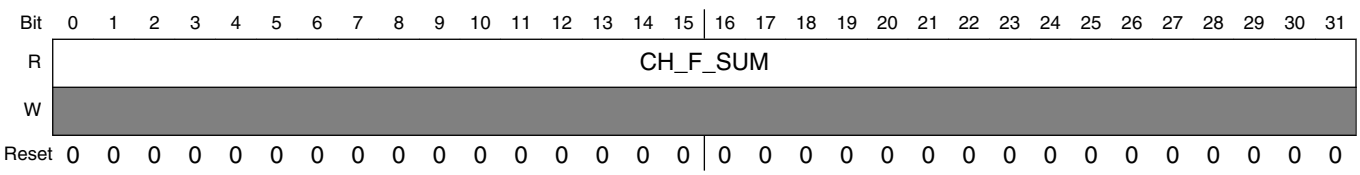


SPT_ACQ_STATUS_E2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_E_BT	Channel E Bit Toggle In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel E input by scanning all samples received from channel E within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.25 Acquisition Status Register F0 (SPT_ACQ_STATUS_F0)

Address: 0h base + 80h offset = 80h



SPT_ACQ_STATUS_F0 field descriptions

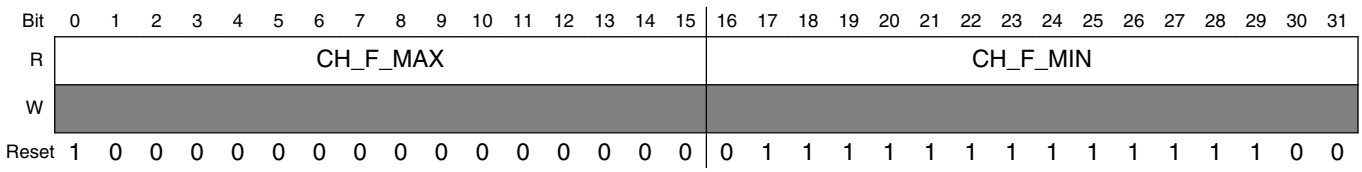
Field	Description
0–31 CH_F_SUM	Channel F Sum This field provides the signed sum of all samples received by channel F within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of

SPT_ACQ_STATUS_F0 field descriptions (continued)

Field	Description
	GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.26 Acquisition Status Register F1 (SPT_ACQ_STATUS_F1)

Address: 0h base + 84h offset = 84h

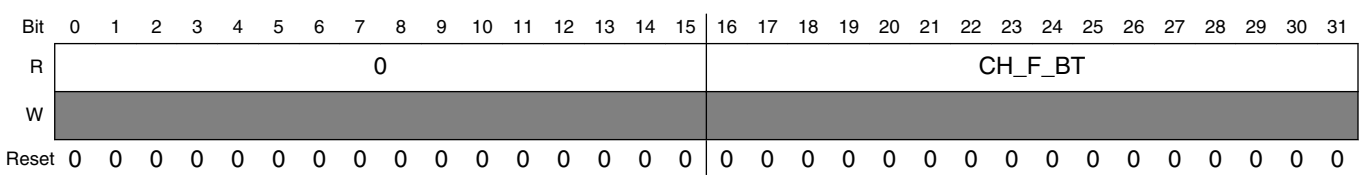


SPT_ACQ_STATUS_F1 field descriptions

Field	Description
0–15 CH_F_MAX	Channel F Maximum This field provides the global (signed) maximum of all samples received from channel F within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_F_MIN	Channel F Minimum This field provides the global (signed) minimum of all samples received from channel F within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.27 Acquisition Status Register F2 (SPT_ACQ_STATUS_F2)

Address: 0h base + 88h offset = 88h



SPT_ACQ_STATUS_F2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_F_BT	Channel F Bit Toggle

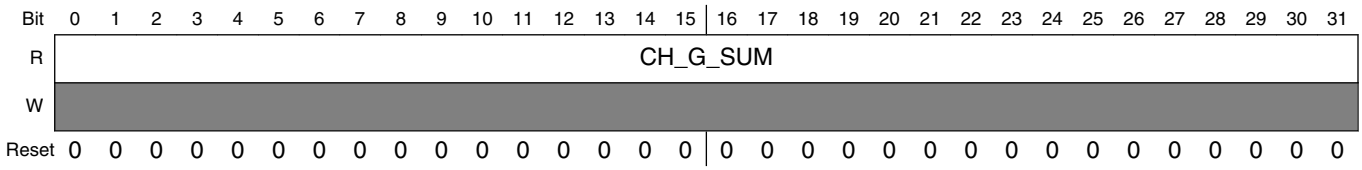
Table continues on the next page...

SPT_ACQ_STATUS_F2 field descriptions (continued)

Field	Description
	In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel F input by scanning all samples received from channel F within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.28 Acquisition Status Register G0 (SPT_ACQ_STATUS_G0)

Address: 0h base + 8Ch offset = 8Ch

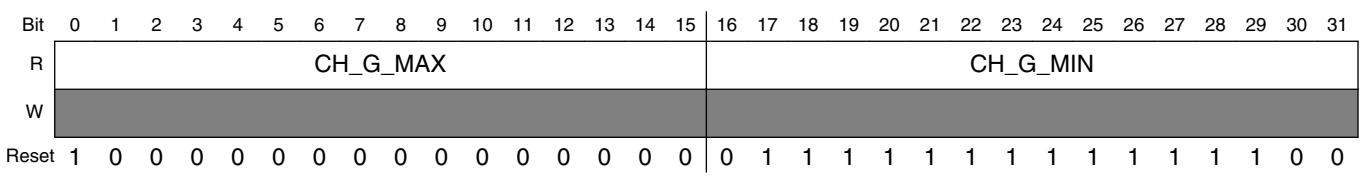


SPT_ACQ_STATUS_G0 field descriptions

Field	Description
0–31 CH_G_SUM	Channel G Sum This field provides the signed sum of all samples received by channel G within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.29 Acquisition Status Register G1 (SPT_ACQ_STATUS_G1)

Address: 0h base + 90h offset = 90h



SPT_ACQ_STATUS_G1 field descriptions

Field	Description
0–15 CH_G_MAX	Channel G Maximum This field provides the global (signed) maximum of all samples received from channel G within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between

Table continues on the next page...

SPT_ACQ_STATUS_G1 field descriptions (continued)

Field	Description
	assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_G_MIN	Channel G Minimum This field provides the global (signed) minimum of all samples received from channel G within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.30 Acquisition Status Register G2 (SPT_ACQ_STATUS_G2)

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CH_G_BT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_G2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_G_BT	Channel G Bit Toggle 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.31 Acquisition Status Register H0 (SPT_ACQ_STATUS_H0)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_H_SUM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_H0 field descriptions

Field	Description
0–31 CH_H_SUM	Channel H Sum This field provides the signed sum of all samples received by channel H within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of

SPT_ACQ_STATUS_H0 field descriptions (continued)

Field	Description
	GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.32 Acquisition Status Register H1 (SPT_ACQ_STATUS_H1)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH_H_MAX																CH_H_MIN															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SPT_ACQ_STATUS_H1 field descriptions

Field	Description
0–15 CH_H_MAX	Channel H Maximum This field provides the global (signed) maximum of all samples received from channel H within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.
16–31 CH_H_MIN	Channel H Minimum This field provides the global (signed) minimum of all samples received from channel H within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence.

45.8.33 Acquisition Status Register H2 (SPT_ACQ_STATUS_H2)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CH_H_BT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_STATUS_H2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CH_H_BT	Channel H Bit Toggle

Table continues on the next page...

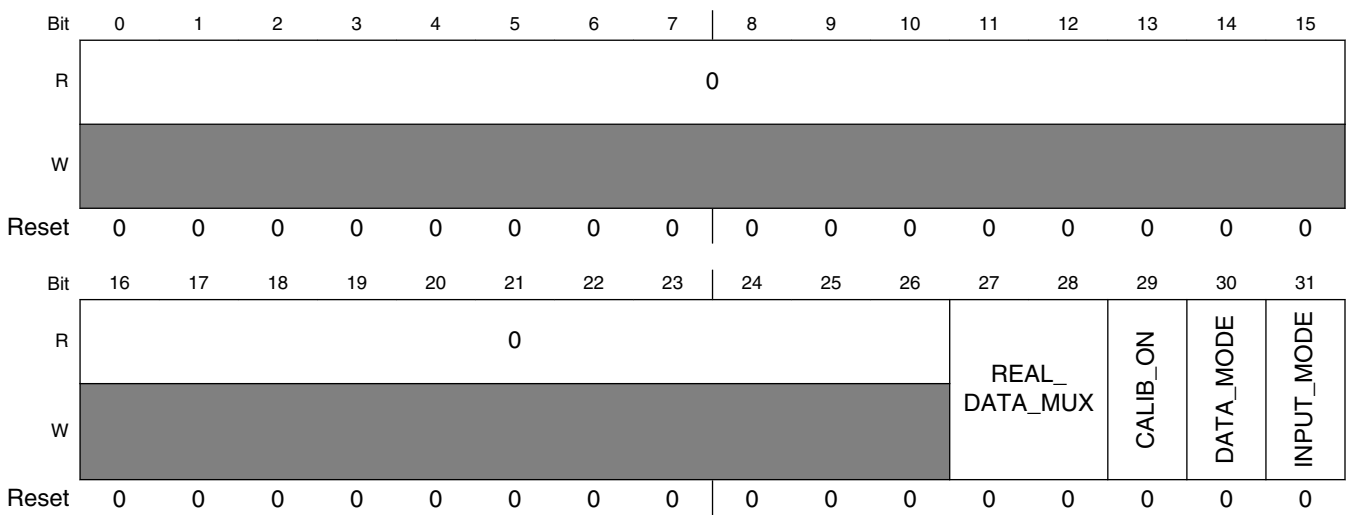
SPT_ACQ_STATUS_H2 field descriptions (continued)

Field	Description
	In this context, toggle is defined as detection of a '1 to 0' transition and a '0 to 1' transition. This field shows toggle status of Channel H input by scanning all samples received from channel H within the acquisition window during the Frame. It is valid to be read only between frames; i.e the period between assertion of GBL_STA[FRM_ACQ_DONE] (or assertion of GBL_STA[CHRP_ACQ_DONE] after last chirp in frame has arrived) and the next RFS occurrence. The input sample toggle status is MSB aligned. Unused LSB bits will be padded with zeros. 0 The corresponding bit did not toggle. 1 The corresponding bit has toggled.

45.8.34 Acquisition MIPICSI2 control register (SPT_ACQ_CSI_CTRL)

This register allows user to configure various control fields as described below only when MIPICSI2 is giving RAW data to Acquisition. No need to configure this for other image formats.

Address: 0h base + A4h offset = A4h



SPT_ACQ_CSI_CTRL field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–28 REAL_DATA_ MUX	Configures the sequence of data when channel 5 is on This register tells the sample rate of each channel data from MIPICSI2. This bit field is used only when calib_on is set to 1. It tells us the sequence of incoming data and repetition of channel 5 data. 00 Channel A-D at 10MSPS and channel E at 10MSPS. RAW14 does not support this mode. 01 Channel A-D at 5MSPS and channel E at 10MSPS.

Table continues on the next page...

SPT_ACQ_CSI_CTRL field descriptions (continued)

Field	Description
	10 Channel A-D at 2.5MSPS and channel E at 10MSPS. 11 Channel A-D at 6.667MSPS and channel E at 13.33MSPS.
29 CALIB_ON	Enables the fifth channel This bit field enables the calibration channel that is the fifth channel (channel E in diagrams). The input mode cannot be tile4 if this bit field is high. 0 calibration channel off 1 calibration channel on
30 DATA_MODE	Defines the type of data. This bit field selects whether the data is Real or complex. If the data is real then only 4 channels are used, else 8 channels are used to store the respective real and imaginary part of each channel. The real and imaginary data come in two consecutive valid clock cycles. 0 Input data is Real only. 1 Input data is complex (Real and Imaginary)
31 INPUT_MODE	Input Mode select This field is used to select input mode as interleaved or tile4 for data from MIPICSI2. Tile4 mode is not supported with data mode as complex. Image data type is always in Interleaved input mode irrespective of this bitfield. 0 Input data is in Tile4 format. Each channel receives 4 consecutive valid data. 1 Input data in Interleaved format. Each channel receives 1 valid data at a time.

45.8.35 Bypass SDMA Control 0 (SPT_SDMA_BYP_CTRL0)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDMA_START_ADDR_BYP																													0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_SDMA_BYP_CTRL0 field descriptions

Field	Description
0–28 SDMA_START_ADDR_BYP	Bypass SDMA Start Address All 32 bits in this register together provide the System RAM address from which Bypass SDMA starts writing ADC packets. The 3 LSB's of this address are restricted to zero, since this address must be a multiple of 8.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.36 Bypass SDMA Control 1 Register (SPT_SDMA_BYP_CTRL1)

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDMA_BUF_LEN_BYP																0					LINE_LEN_BYP									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_SDMA_BYP_CTRL1 field descriptions

Field	Description
0–15 SDMA_BUF_ LEN_BYP	Bypass SDMA Buffer Length This field provides the number of chirps in which samples received can be stored in System RAM continuously without being over-written. Once this is done, wrap-around occurs and previous samples in System RAM will be over-written. The default value of this field is 3. A value of 0 is invalid and hence should not be programmed in this field. For example, if this field is set to 2, then samples received from first 2 chirps are stored in SRAM one after the other depending on the active SDMA addressing mode. The SDMA write address resets to SPT_SDMA_BYP_CTRL0[SDMA_START_ADDR_BYP] from 3'rd chirp onwards, over-writing samples from previous two chirps with samples from 3'rd and 4'th chirps respectively.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 LINE_LEN_BYP	Line Length Bypass This field is utilized when SDMA_CTRL1[TRF_MOD] = 1; i.e. 2D Mode is selected. The value of line length is given by (LINE_LEN_BYP + 1)*8 Hence configuring this field to 0 results in line length of 8. Similarly a value of 31 corresponds to line length of 256. Maximum value of 2047 corresponds to line length of 16384.

45.8.37 Acquisition Bypass Control Register (SPT_ACQ_BYP_CTRL)

This register allows user to configure various control fields as described below for bypass mode of ACQ in case of MIPICSI2 as input.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NUM_SMPL_BYP																0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_ACQ_BYP_CTRL field descriptions

Field	Description
0–15 NUM_SMPL_ BYP	Number of Samples in a Chirp This value is given by (NUM_SMPL_BYP + 1)*8 Hence configuring this field to 0 results in 8 samples per chirp. Similarly the default value of 31 corresponds to 256 samples per chirp. Maximum value of 65,535 corresponds to 524,288 samples per chirp. This field is programmed when the SPT_ACQ_CSI_CTRL[CALIB_ON] is enabled or the data type from MIPICSI2 is not RAW. This field is invalid if ACQ_GBL_CTRL_0[INF_SMPLS] is asserted.
16–31 Reserved	reserved This field is reserved. This read-only field is reserved and always has the value 0.

45.8.38 Program Start Addr Register (SPT_CS_PG_ST_ADDR)

Address: 0h base + C0h offset = C0h

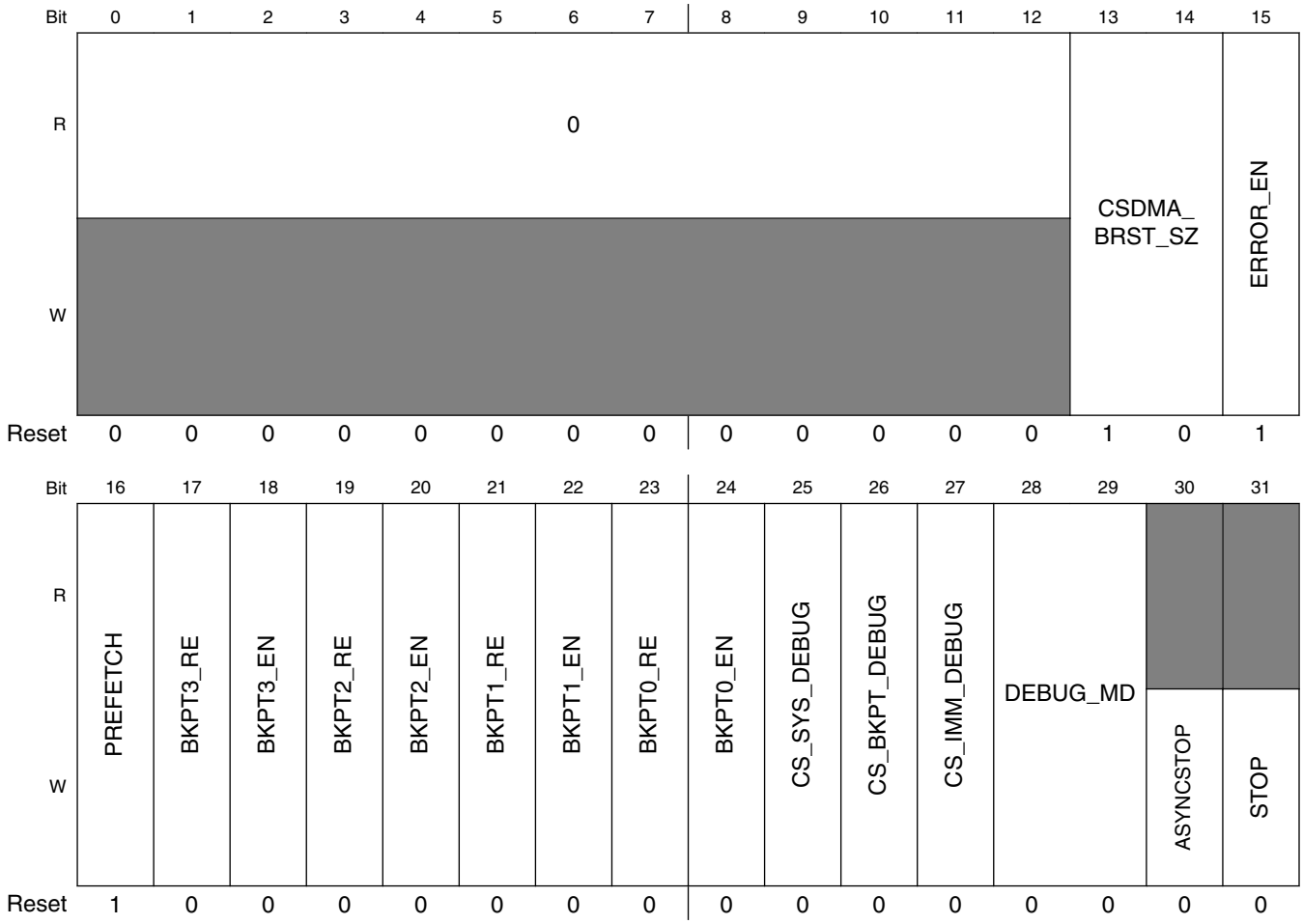
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PG_ST_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_PG_ST_ADDR field descriptions

Field	Description
0–31 PG_ST_ADDR	Program start address Gives the starting address for program execution. This is the fetch address of instructions from system memory. This address should be a multiple of 8. This address should be written before setting the SPT_GBL_CTRL[PG_ST_CTRL] field.

45.8.39 Mode Control Register (SPT_CS_MODE_CTRL)

Address: 0h base + C4h offset = C4h



SPT_CS_MODE_CTRL field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–14 CSDMA_BRST_SZ	Command Sequencer DMA Burst Size This field allows the user to choose the size of the command sequencer DMA burst 00 The burst size is INCR4 01 The burst size is INCR8 1X The burst size is INCR16
15 ERROR_EN	Error Enable If this bit is set then a command error causes the command sequencer to go to the ASYNCSTOP state. If it is reset then the command sequencer ignores the error and moves to the subsequent instruction. However it should be noted that the loop error (more than 4 loops), next error (NEXT instruction fired

Table continues on the next page...

SPT_CS_MODE_CTRL field descriptions (continued)

Field	Description
	before LOOP instruction), iteration error (loop of zero iteration) and opcode error (illegal opcode value) always cause the sequencer to go to the ASYNCSTOP state. The ERROR_EN bit has no effect on these errors.
16 PREFETCH	Early Pre-fetch Enable This bit controls the early pre-fetch option of the command sequencer. 0 Early pre-fetch option disabled. 1 Early pre-fetch option enabled.
17 BKPT3_RE	Breakpoint 3 remains enabled If set then the breakpoint 3 remains enabled after sequencer has hit this breakpoint, else it is disabled.
18 BKPT3_EN	Breakpoint 3 enabled If set then the breakpoint 3 is enabled
19 BKPT2_RE	Breakpoint 2 remains enabled If set then the breakpoint 2 remains enabled after sequencer has hit this breakpoint, else it is disabled.
20 BKPT2_EN	Breakpoint 2 enabled If set then the breakpoint 2 is enabled
21 BKPT1_RE	Breakpoint 1 remains enabled If set then the breakpoint 1 remains enabled after sequencer has hit this breakpoint, else it is disabled.
22 BKPT1_EN	Breakpoint 1 enabled If set then the breakpoint 1 is enabled
23 BKPT0_RE	Breakpoint 0 remains enabled If set then the breakpoint 0 remains enabled after sequencer has hit this breakpoint, else it is disabled.
24 BKPT0_EN	Breakpoint 0 enabled If set then the breakpoint 0 is enabled
25 CS_SYS_ DEBUG	System debug enable If set along with the 'system debug' input signal it causes the command sequencer to go into the DEBUG state at the end of the current instruction. If not set and the 'system debug' input signal is HIGH, then the command sequencer just ignores the signal.
26 CS_BKPT_ DEBUG	Breakpoint debug enable If set then sequencer goes from RUN state to DEBUG state when an enabled breakpoint is hit
27 CS_IMM_ DEBUG	IMM DEBUG enable If set then the sequencer goes from RUN state to the DEBUG state at the end of execution of current instruction.
28–29 DEBUG_MD	DEBUG MODE control This field decides the mode of debug when the command sequencer enters the DEBUG state. 00 - halt mode. The command sequencer remains halted. 01 - step once mode. The sequencer steps one instruction ahead at a time. 10 - step jump mode. The sequencer keeps executing instructions till it hits an enabled breakpoint. 11 - instruction jamming mode. The sequencer enters the instruction jamming mode.

Table continues on the next page...

SPT_CS_MODE_CTRL field descriptions (continued)

Field	Description
30 ASYNCSTOP	Asynchronous STOP bit If set it causes the sequencer (either in RUN, DEBUG or WAIT state) to go to the ASYNCSTOP state immediately. It is set by SW and is reset internally in the next cycle.
31 STOP	STOP bit If set it causes the sequencer (either in RUN, DEBUG or WAIT state) to go to the STOP state at the end of execution of the present instruction. It is set by SW and is reset internally in the next cycle.

45.8.40 Watchdog Counter Register (SPT_CS_WD_COUNT)

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WD_COUNT																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_WD_COUNT field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 WD_COUNT	Watchdog timer count Gives the watchdog timer count. The counter value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read of the counter value is not possible. The counter value is refreshed every 32 clock cycles of the SPT domain.

45.8.41 Breakpoint0 Addr Register (SPT_CS_BKPT0_ADDR)

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BKPT0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_BKPT0_ADDR field descriptions

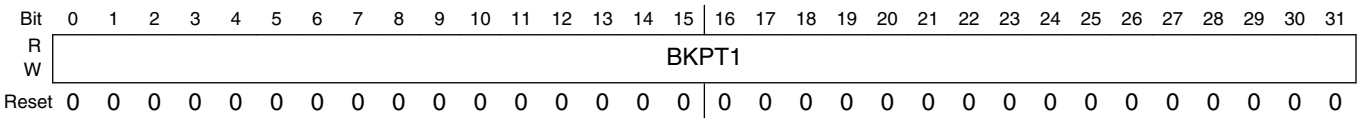
Field	Description
0–31 BKPT0	Breakpoint 0 address Gives the breakpoint 0 address. It is the absolute system memory address. This address should be written by SW either outside the step jump mode or before clearing the SPT_CS_STATUS0[STEP_JUMP_OVR] field. If it is required to change the breakpoint address on the fly in the step jump mode then SW must first

SPT_CS_BKPT0_ADDR field descriptions (continued)

Field	Description
	take the command sequencer to the halt mode, change the address and then go back to the step jump mode.

45.8.42 Breakpoint1 Addr Register (SPT_CS_BKPT1_ADDR)

Address: 0h base + D0h offset = D0h

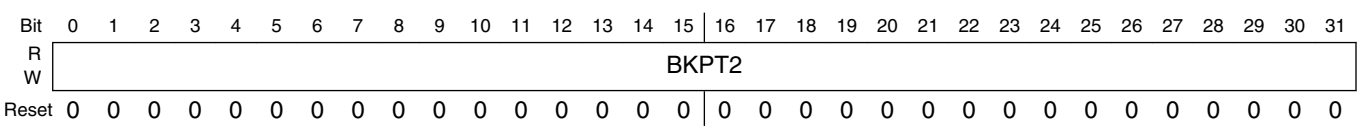


SPT_CS_BKPT1_ADDR field descriptions

Field	Description
0–31 BKPT1	Breakpoint 1 address Gives the breakpoint 1 address. It is the absolute system memory address. This address should be written by SW either outside the step jump mode or before clearing the SPT_CS_STATUS0[STEP_JUMP_OVR] field. If it is required to change the breakpoint address on the fly in the step jump mode then SW must first take the command sequencer to the halt mode, change the address and then go back to the step jump mode.

45.8.43 Breakpoint2 Addr Register (SPT_CS_BKPT2_ADDR)

Address: 0h base + D4h offset = D4h



SPT_CS_BKPT2_ADDR field descriptions

Field	Description
0–31 BKPT2	Breakpoint 2 address Gives the breakpoint 2 address. It is the absolute system memory address. This address should be written by SW either outside the step jump mode or before clearing the SPT_CS_STATUS0[STEP_JUMP_OVR] field. If it is required to change the breakpoint address on the fly in the step jump mode then SW must first take the command sequencer to the halt mode, change the address and then go back to the step jump mode.

45.8.44 Breakpoint3 Addr Register (SPT_CS_BKPT3_ADDR)

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BKPT3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_BKPT3_ADDR field descriptions

Field	Description
0–31 BKPT3	Breakpoint 3 address Gives the breakpoint 3 address. It is the absolute system memory address. This address should be written by SW either outside the step jump mode or before clearing the SPT_CS_STATUS0[STEP_JUMP_OVR] field. If it is required to change the breakpoint address on the fly in the step jump mode then SW must first take the command sequencer to the halt mode, change the address and then go back to the step jump mode.

45.8.45 Jamming Instruction0 Register (SPT_CS_JAM_INST0)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JAM_INST_127_96																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_JAM_INST0 field descriptions

Field	Description
0–31 JAM_INST_127_ 96	Jamming instruction bits 127:96 Bits 127:96 of the 128-bit register which holds the instruction to be executed in the instruction jamming mode. This field should be written before clearing the SPT_CS_STATUS0[JAM_OVR] field.

45.8.46 Jamming Instruction1 Register (SPT_CS_JAM_INST1)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JAM_INST_95_64																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_JAM_INST1 field descriptions

Field	Description
0–31 JAM_INST_95_64	Jamming instruction bits 95:64 Bits 95:64 of the 128-bit register which holds the instruction to be executed in the instruction jamming mode. This field should be written before clearing the SPT_CS_STATUS0[JAM_OVR] field.

45.8.47 Jamming Instruction2 Register (SPT_CS_JAM_INST2)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JAM_INST_63_32																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_JAM_INST2 field descriptions

Field	Description
0–31 JAM_INST_63_32	Jamming instruction bits 63:32 Bits 63:32 of the 128-bit register which holds the instruction to be executed in the instruction jamming mode. This field should be written before clearing the SPT_CS_STATUS0[JAM_OVR] field.

45.8.48 Jamming Instruction3 Register (SPT_CS_JAM_INST3)

Address: 0h base + E8h offset = E8h

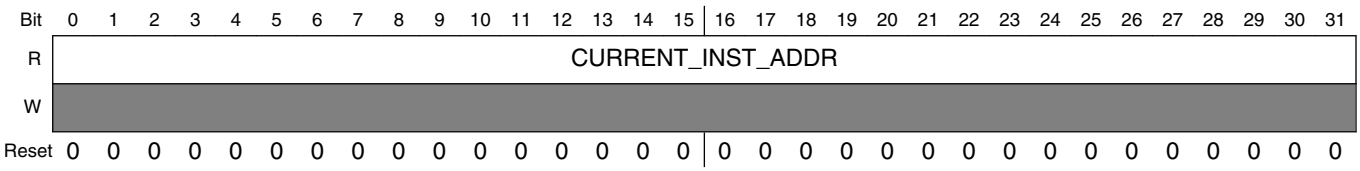
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JAM_INST_31_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_JAM_INST3 field descriptions

Field	Description
0–31 JAM_INST_31_0	Jamming instruction bits 31:0 Bits 31:0 of the 128-bit register which holds the instruction to be executed in the instruction jamming mode. This field should be written before clearing the SPT_CS_STATUS0[JAM_OVR] field.

45.8.49 Current Instruction Addr Register (SPT_CS_CURR_INST_ADDR)

Address: 0h base + ECh offset = ECh

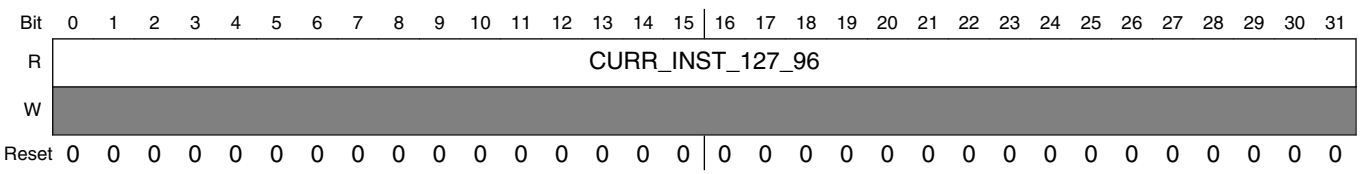


SPT_CS_CURR_INST_ADDR field descriptions

Field	Description
0–31 CURRENT_ INST_ADDR	Current instruction pointer Gives the address (corresponding to the system memory) of the instruction being executed. The current instruction pointer value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.

45.8.50 Current Instruction0 Register (SPT_CS_CURR_INST0)

Address: 0h base + F0h offset = F0h

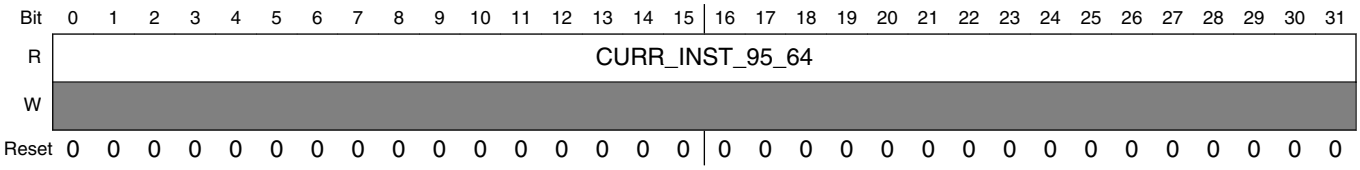


SPT_CS_CURR_INST0 field descriptions

Field	Description
0–31 CURR_INST_ 127_96	Current instruction bits 127:96 Bits 127:96 of the 128-bit status register which gives the actual instruction being currently executed. It should be noted that SW needs to perform 4 read operations on SPT_CS_CURR_INST0, SPT_CS_CURR_INST1, SPT_CS_CURR_INST2 and SPT_CS_CURR_INST3 to get the entire instruction's contents. Thus a synchronous read out is not guaranteed unless the command sequencer is in the HALT mode of the DEBUG state. Also it must be noted that the current instruction value propagates from the SPT domain to the slower peripheral interface and hence also a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.

45.8.51 Current Instruction1 Register (SPT_CS_CURR_INST1)

Address: 0h base + F4h offset = F4h

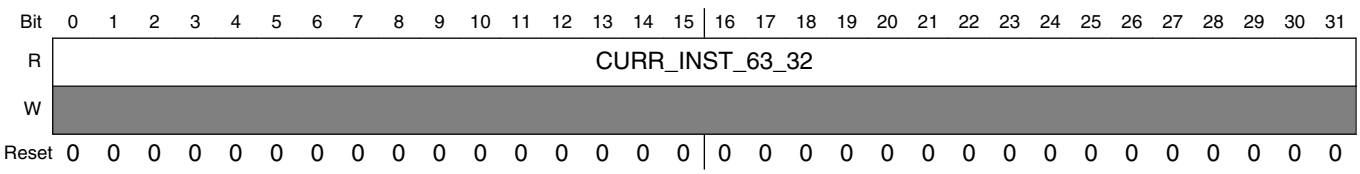


SPT_CS_CURR_INST1 field descriptions

Field	Description
0–31 CURR_INST_ 95_64	Current instruction bits 95:64 Bits 95:64 of the 128-bit status register which gives the actual instruction being currently executed.

45.8.52 Current Instruction2 Register (SPT_CS_CURR_INST2)

Address: 0h base + F8h offset = F8h

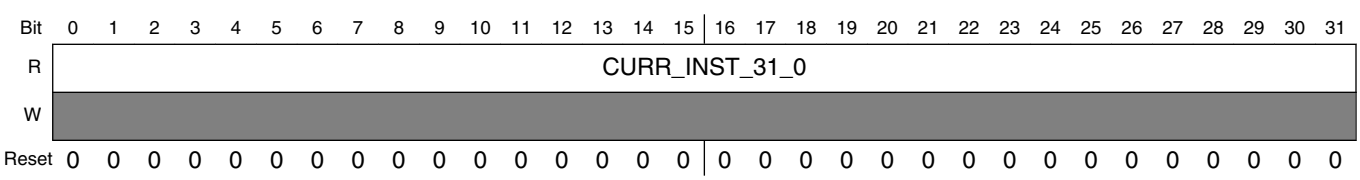


SPT_CS_CURR_INST2 field descriptions

Field	Description
0–31 CURR_INST_ 63_32	Current instruction bits 63:32 Bits 63:32 of the 128-bit status register which gives the actual instruction being currently executed.

45.8.53 Current Instruction3 Register (SPT_CS_CURR_INST3)

Address: 0h base + FCh offset = FCh



SPT_CS_CURR_INST3 field descriptions

Field	Description
0–31 CURR_INST_ 31_0	Current instruction bits 31:0 Bits 31:0 of the 128-bit status register which gives the actual instruction being currently executed.

45.8.54 Loop Counter 0 and 1 Register (SPT_CS_LOOPCNTR01)

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LOOP_CNTR1																LOOP_CNTR0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_LOOPCNTR01 field descriptions

Field	Description
0–15 LOOP_CNTR1	Loop count of counter 1 Gives the count of the loop counter 1. The counter counts down to zero. The counter value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.
16–31 LOOP_CNTR0	Loop count of counter 0 Gives the count of the loop counter 0. The counter counts down to zero. The counter value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.

45.8.55 Loop Counter 2 and 3 Register (SPT_CS_LOOPCNTR23)

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LOOP_CNTR3																LOOP_CNTR2															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_LOOPCNTR23 field descriptions

Field	Description
0–15 LOOP_CNTR3	Loop count of counter 3 Gives the count of the loop counter 3. The counter counts down to zero. The counter value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.

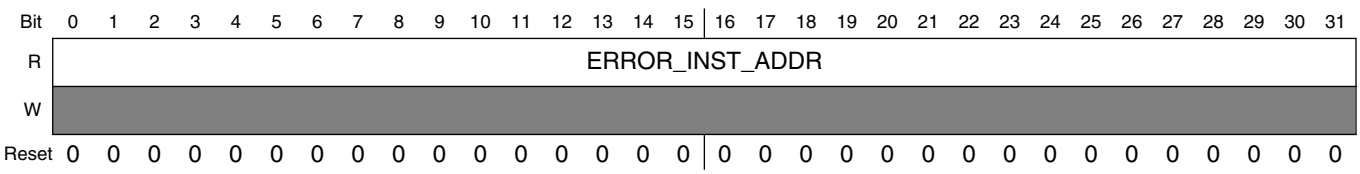
Table continues on the next page...

SPT_CS_LOOPCNTR23 field descriptions (continued)

Field	Description
16–31 LOOP_CNTR2	Loop count of counter 2 Gives the count of the loop counter 2. The counter counts down to zero. The counter value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.

45.8.56 Error Instruction Addr Register (SPT_CS_ERR_INST_ADDR)

Address: 0h base + 108h offset = 108h

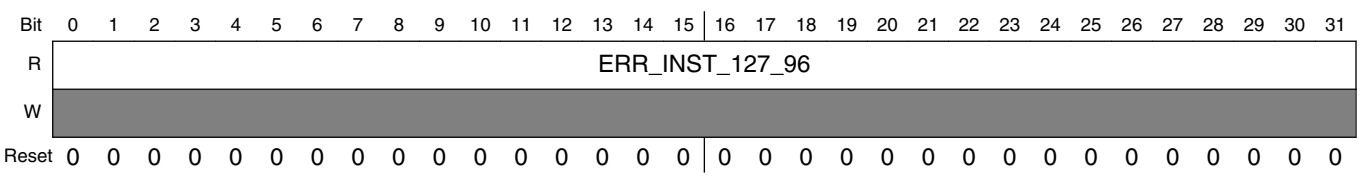


SPT_CS_ERR_INST_ADDR field descriptions

Field	Description
0–31 ERROR_INST_ ADDR	Error instruction pointer Gives the address (corresponding to the system memory) of the instruction which caused the command error.

45.8.57 Error Instruction 0 Register (SPT_CS_ERR_INST0)

Address: 0h base + 10Ch offset = 10Ch

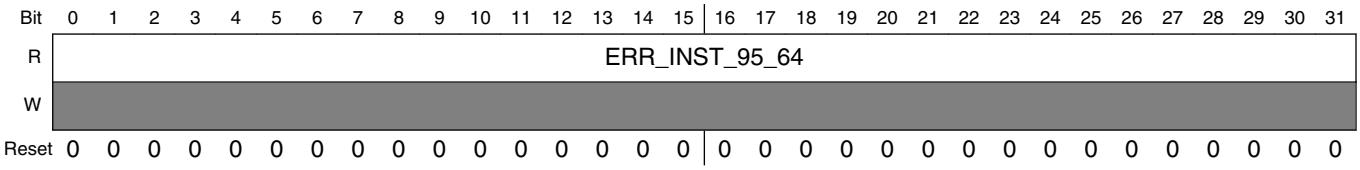


SPT_CS_ERR_INST0 field descriptions

Field	Description
0–31 ERR_INST_127_ 96	Error instruction bits 127:96 Bits 127:96 of the 128-bit status register which gives the actual instruction which caused the error.

45.8.58 Error Instruction 1 Register (SPT_CS_ERR_INST1)

Address: 0h base + 110h offset = 110h

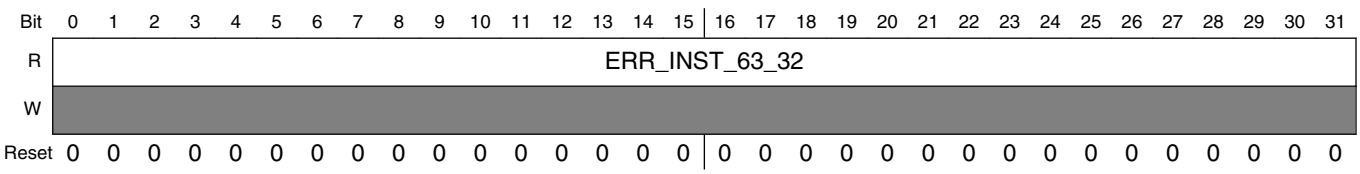


SPT_CS_ERR_INST1 field descriptions

Field	Description
0–31 ERR_INST_95_64	Error instruction bits 95:64 Bits 95:64 of the 128-bit status register which gives the actual instruction which caused the error.

45.8.59 Error Instruction 2 Register (SPT_CS_ERR_INST2)

Address: 0h base + 114h offset = 114h

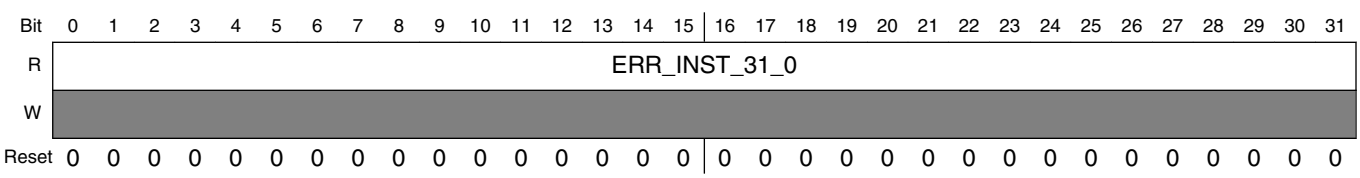


SPT_CS_ERR_INST2 field descriptions

Field	Description
0–31 ERR_INST_63_32	Error instruction bits 63:32 Bits 63:32 of the 128-bit status register which gives the actual instruction which caused the error.

45.8.60 Error Instruction 3 Register (SPT_CS_ERR_INST3)

Address: 0h base + 118h offset = 118h

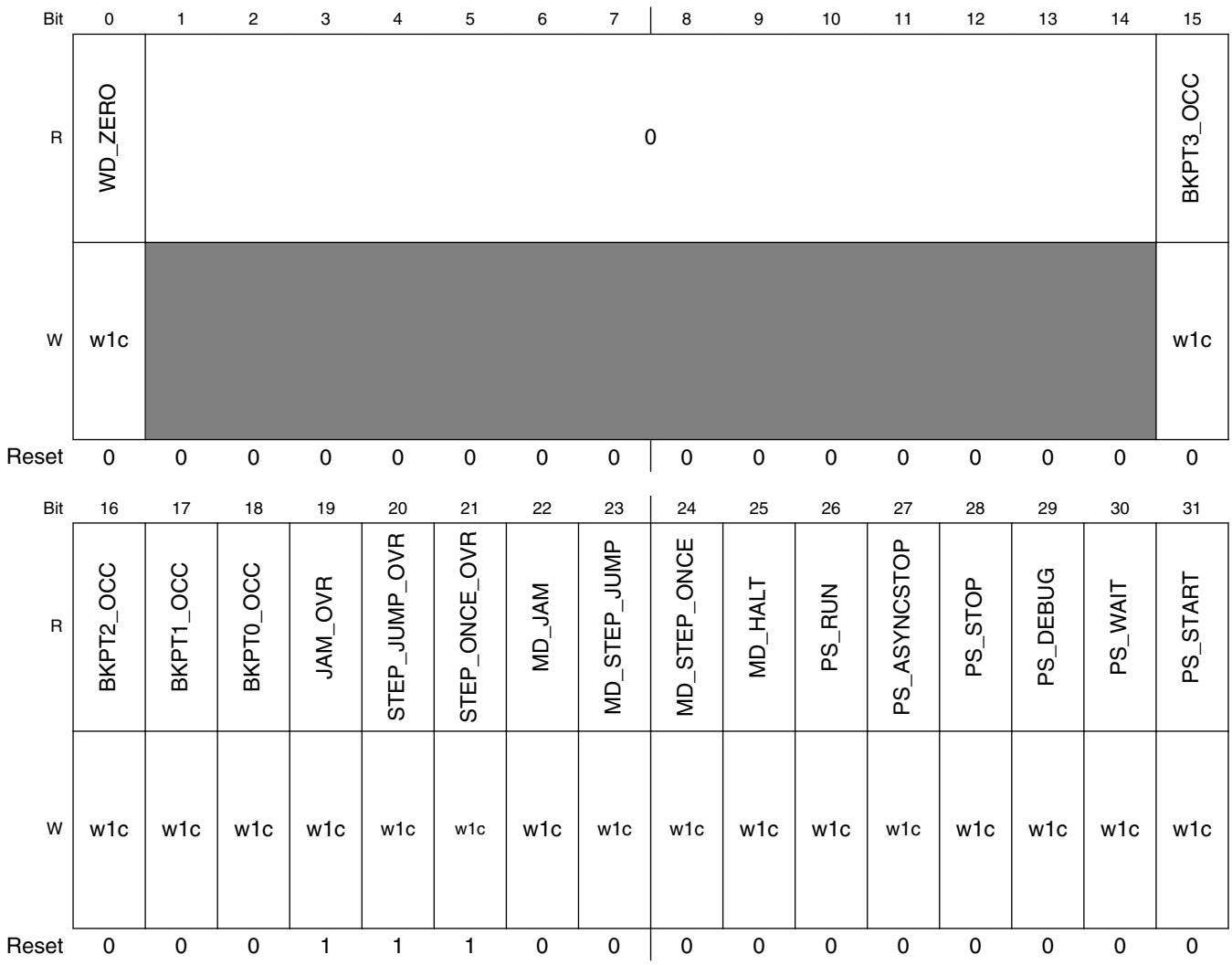


SPT_CS_ERR_INST3 field descriptions

Field	Description
0–31 ERR_INST_31_0	Error instruction bits 31:0 Bits 31:0 of the 128-bit status register which gives the actual instruction which caused the error.

45.8.61 General Status 0 Register (SPT_CS_STATUS0)

Address: 0h base + 11Ch offset = 11Ch



SPT_CS_STATUS0 field descriptions

Field	Description
0 WD_ZERO	Watchdog timer reached zero This field is set when the watchdog timer goes to zero in the count mode. Is internally reset in the SETUP state

Table continues on the next page...

SPT_CS_STATUS0 field descriptions (continued)

Field	Description
1–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BKPT3_OCC	'Breakpoint 3 hit' indication Set when the command sequencer hits the breakpoint 3 in the STEP JUMP mode and there is no ongoing asynchronous PDMA instruction. Is internally reset in the SETUP state
16 BKPT2_OCC	'Breakpoint 2 hit' indication Set when the command sequencer hits the breakpoint 2 in the STEP JUMP mode and there is no ongoing asynchronous PDMA instruction. Is internally reset in the SETUP state
17 BKPT1_OCC	'Breakpoint 1 hit' indication Set when the command sequencer hits the breakpoint 1 in the STEP JUMP mode and there is no ongoing asynchronous PDMA instruction. Is internally reset in the SETUP state
18 BKPT0_OCC	'Breakpoint 0 hit' indication Set when the command sequencer hits the breakpoint 0 in the STEP JUMP mode and there is no ongoing asynchronous PDMA instruction. Is internally reset in the SETUP state
19 JAM_OVR	Jamming mode instruction completion indication This field is set when program execution starts. When a '1' is written to it then the field gets reset and if the command sequencer is in the instruction jamming mode then a new instruction is executed. It gets set when an instruction is completed in the instruction jamming mode.
20 STEP_JUMP_ OVR	STEP JUMP mode break indication This field is set when program execution starts. When a '1' is written to it then the field gets reset and if the command sequencer is in STEP JUMP mode it starts executing instructions. It gets set in the STEP JUMP mode when a breakpoint is hit (at the end of execution of the instruction) and there is no ongoing asynchronous PDMA instruction
21 STEP_ONCE_ OVR	STEP ONCE mode instruction completion indication This field is set when program execution starts. When a '1' is written to it then the field gets reset and if the command sequencer is in the STEP ONCE mode it executes a single instruction. It gets set when an instruction is completed in the STEP ONCE mode.
22 MD_JAM	INSTRUCTION JAMMING mode indicator Set when the command sequencer enters the instruction jamming mode. Is internally reset in the SETUP state
23 MD_STEP_ JUMP	STEP JUMP mode indicator Is set when the sequencer enters the DEBUG state's STEP JUMP mode. Is internally reset in the SETUP state
24 MD_STEP_ ONCE	STEP ONCE mode indicator Is set when the sequencer enters the DEBUG state's STEP ONCE mode. Is internally reset in the SETUP state
25 MD_HALT	HALT MODE indicator Is set when the sequencer enters the DEBUG state's halt mode. Is internally reset in the SETUP state
26 PS_RUN	RUN state indicator

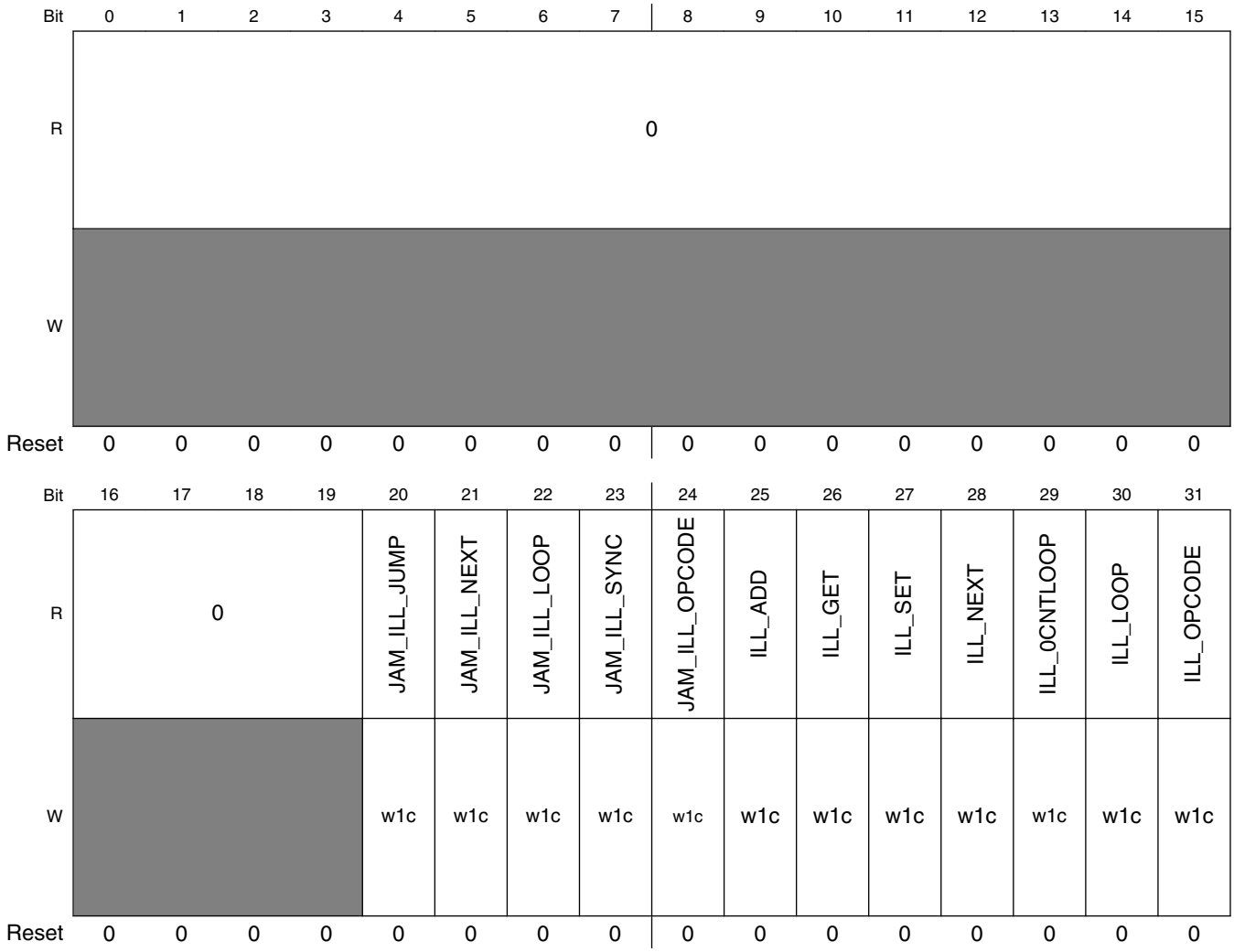
Table continues on the next page...

SPT_CS_STATUS0 field descriptions (continued)

Field	Description
	Set when the command sequencer enters the RUN state. Is internally reset in the SETUP state
27 PS_ ASYNCSTOP	ASYNCSTOP state indicator Set when the command sequencer enters the ASYNCSTOP state. Is internally reset in the SETUP state
28 PS_STOP	STOP state indicator Set when the command sequencer enters the STOP state. Is internally reset in the SETUP state
29 PS_DEBUG	DEBUG state indicator Set when the command sequencer enters the DEBUG state. Is internally reset in the SETUP state
30 PS_WAIT	WAIT state indicator Set when the command sequencer enters the WAIT state. Is internally reset in the SETUP state
31 PS_START	START state indicator Set when the command sequencer enters the START state. Is internally reset in the SETUP state

45.8.62 General Status 1 Register (SPT_CS_STATUS1)

Address: 0h base + 120h offset = 120h



SPT_CS_STATUS1 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 JAM_ILL_JUMP	Illegal JUMP in jamming mode Indicates that 'JUMP' instruction was fired, which is illegal in the instruction jamming mode. Is internally reset in the SETUP state.
21 JAM_ILL_NEXT	Illegal NEXT in jamming mode Indicates that 'NEXT' instruction was fired, which is illegal in the instruction jamming mode. Is internally reset in the SETUP state.

Table continues on the next page...

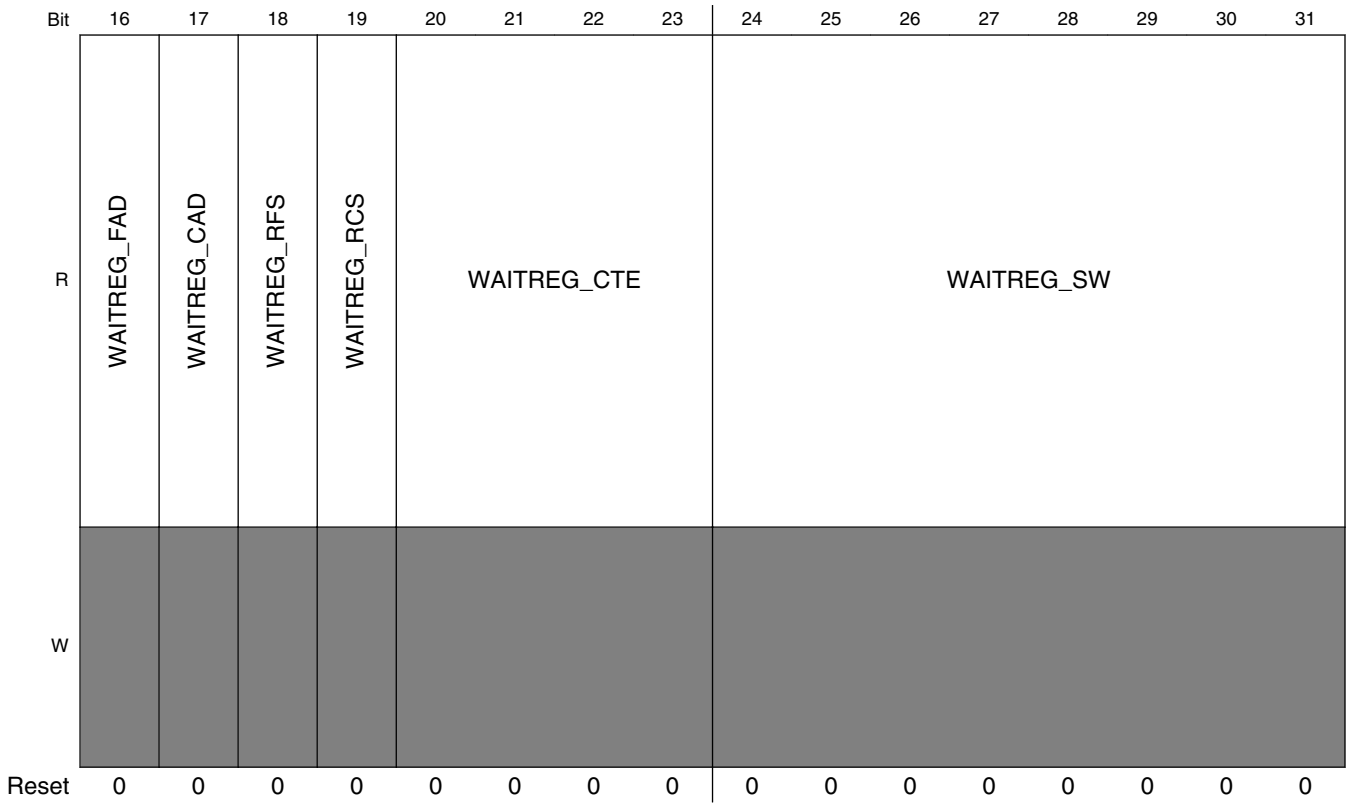
SPT_CS_STATUS1 field descriptions (continued)

Field	Description
22 JAM_ILL_LOOP	Illegal LOOP in jamming mode Indicates that 'LOOP' instruction was fired, which is illegal in the instruction jamming mode. Is internally reset in the SETUP state.
23 JAM_ILL_SYNC	Illegal SYNC in jamming mode Indicates that 'SYNC' instruction was fired, which is illegal in the instruction jamming mode. Is internally reset in the SETUP state.
24 JAM_ILL_ OPCODE	Illegal opcode in jamming mode Indicates that an illegal instruction opcode was fired while in the instruction jamming mode. Is internally reset in the SETUP state.
25 ILL_ADD	Illegal ADD Indicates that an illegal 'ADD' or illegal 'SUB' or illegal 'CMP' or illegal 'SEL' command instruction was fired. Is internally reset in the SETUP state. The user can also check the SPT_CS_ERR_INST_ADDR register to see which instruction caused the error. It is not set in the jamming mode in debug state.
26 ILL_GET	Illegal GET Indicates that an illegal 'GET' instruction was fired. It is not set in the jamming mode in debug state. Is internally reset in the SETUP state.
27 ILL_SET	Illegal SET Indicates that an illegal 'SET' instruction was fired. It is not set in the jamming mode in debug state. Is internally reset in the SETUP state.
28 ILL_NEXT	Illegal NEXT Indicates that a NEXT instruction was fired though there was no loop in progress. Is internally reset in the SETUP state. It is not set in the jamming mode in debug state.
29 ILL_0CNTLOOP	Illegal loop count error Indicates that a LOOP instruction with zero iterations was fired. Is internally reset in the SETUP state. It is not set in the jamming mode in debug state.
30 ILL_LOOP	illegal LOOP Indicates that a LOOP instruction was fired to start a fifth loop, and only four are allowed. Is internally reset in the SETUP state. It is not set in the jamming mode in debug state.
31 ILL_OPCODE	Illegal opcode Indicates that an illegal instruction opcode was fired. Is internally reset in the SETUP state. It is not set in the jamming mode in debug state.

45.8.63 General Status2 Register (SPT_CS_STATUS2)

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					WD_WAITREG_TR		WD_WAITREG_RFS	WD_WAITREG_RCS	WD_WAITREG_CTE				WAITREG_TR		WAITREG_PXD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SPT_CS_STATUS2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–6 WD_WAITREG_TR	Event trigger type for WATCHDOG timer Specifies whether the awaited event is level-0 triggered, level-1 triggered, posedge triggered or negedge triggered. 00 – level-0 triggered; 01 – level-1 triggered; 10 – posedge triggered; 11 – negedge triggered;
7 WD_WAITREG_RFS	Watchdog waits for cte_rfs or csi_vsync signal If set then indicates that the watchdog waits for event on cte_rfs or csi_vsync signal. At any given time only one bit of SPT_CS_STATUS2[WD_WAITREG_RFS], SPT_CS_STATUS2[WD_WAITREG_RCS] and SPT_CS_STATUS2[WD_WAITREG_CTE] is set.
8 WD_WAITREG_RCS	Watchdog waits for cte_rcs or csi_hsync signal If set then indicates that the watchdog waits for event on cte_rcs or csi_hsync signal
9–12 WD_WAITREG_CTE	Watchdog waits for CTE
13–14 WAITREG_TR	Event trigger type Specifies whether the awaited event is level-0 triggered, level-1 triggered, posedge triggered or negedge triggered. 00 – level-0 triggered; 01 – level-1 triggered; 10 – posedge triggered; 11 – negedge triggered;
15 WAITREG_PXD	WAIT status for 'pdma transfer done' signal

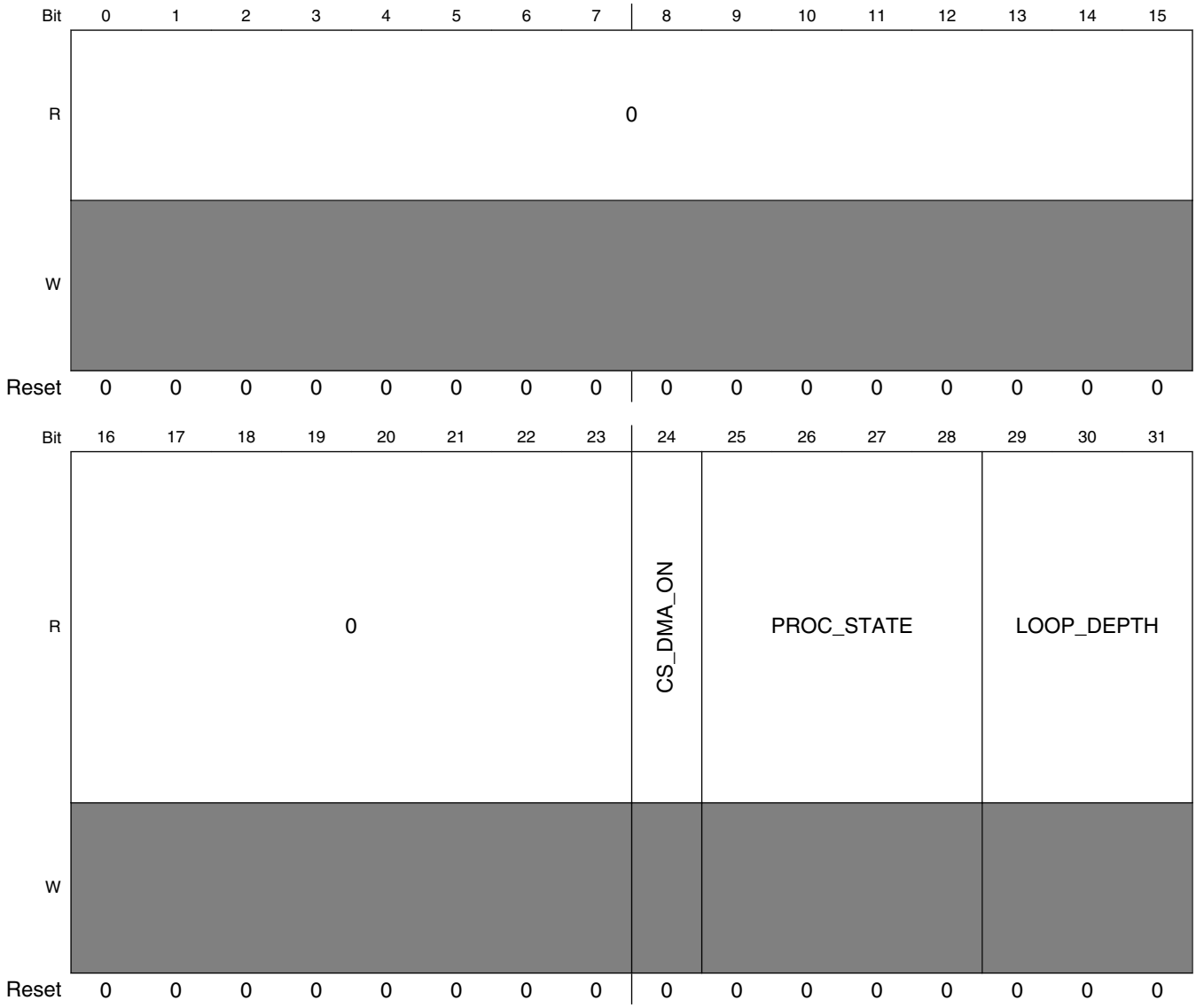
Table continues on the next page...

SPT_CS_STATUS2 field descriptions (continued)

Field	Description
	If set then indicates that the sequencer is waiting for event on 'pdma transfer done' signal. At any given time only 1 bit of SPT_CS_STATUS2[WAITREG_PXD], SPT_CS_STATUS2[WAITREG_FAD], SPT_CS_STATUS2[WAITREG_CAD], SPT_CS_STATUS2[WAITREG_RFS], SPT_CS_STATUS2[WAITREG_RCS], SPT_CS_STATUS2[WAITREG_CTE] or SPT_CS_STATUS2[WAITREG_SW] is set.
16 WAITREG_FAD	WAIT status for 'frame acquisition done' signal If set then indicates that the sequencer is waiting for event on 'frame acquisition done' signal.
17 WAITREG_CAD	WAIT status for 'chirp acquisition done' signal If set then indicates that the sequencer is waiting for event on 'chirp acquisition done' signal.
18 WAITREG_RFS	WAIT status for cte_rfs If set then indicates that the sequencer is waiting for event on cte_rfs or csi_vsync signal.
19 WAITREG_RCS	WAIT status for cte_rcs If set then indicates that the sequencer is waiting for event on cte_rcs or csi_hsync signal.
20–23 WAITREG_CTE	WAIT status for CTE event If any bit is set then it indicates that sequencer is waiting for the corresponding bit of cte_evt[3:0] 1000- Waits for event on cte_evt[3] 0100- Waits for event on cte_evt[2] 0010- Waits for event on cte_evt[1] 0001- Waits for event on cte_evt[0]
24–31 WAITREG_SW	WAIT status for SW-enforced event If any bit is set then it indicates that sequencer is waiting for the corresponding bit of the SW-enforced event 00000001- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[0]] 00000010- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[1]] 00000100- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[2]] 00001000- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[3]] 00010000- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[4]] 00100000- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[5]] 01000000- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[6]] 10000000- Waits for event triggered on SPT_CS_SW_EVTREG[SW_EVTREG[7]]

45.8.64 General Status 3 Register (SPT_CS_STATUS3)

Address: 0h base + 128h offset = 128h



SPT_CS_STATUS3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CS_DMA_ON	Command sequencer DMA ON Provides the status that command sequencer is accessing the AHB bus
25–28 PROC_STATE	Processing state

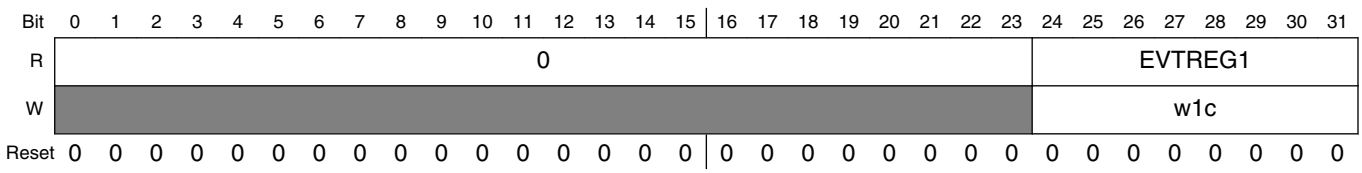
Table continues on the next page...

SPT_CS_STATUS3 field descriptions (continued)

Field	Description
	Gives the present processing state of the command sequencer FSM. RST = 0000b. START = 0001b. SETUP = 0010b. RUN = 0011b. WAIT = 0100b. DEBUG = 0101b. STOP = 0110b. ASYNCSTOP = 0111b. The processing state value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. It is refreshed every 32 clock cycles of the SPT domain.
29–31 LOOP_DEPTH	Loop depth Gives the depth of the command sequencer loop being executed. A value of 0 means that no loop is being executed. The loop depth value propagates from the SPT domain to the slower peripheral interface and thus a synchronous read is not possible. The loop depth value is refreshed every 32 clock cycles of the SPT domain.

45.8.65 EVT1 Status Register (SPT_CS_EVTREG1)

Address: 0h base + 12Ch offset = 12Ch

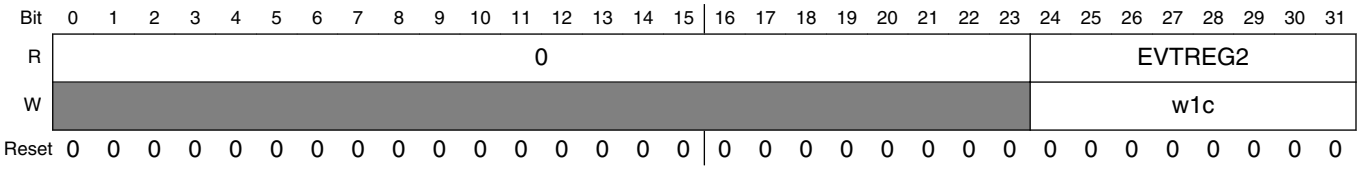


SPT_CS_EVTREG1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 EVTREG1	EVT1 register Each bit corresponds to an event. Whenever an EVT command occurs a bit corresponding to the specified event (given in the command word by the field name 'EV') is set when 'EV_LVL' field in the command word is 1 or reset when 'EV_LVL' is 0. This field is internally reset in the SETUP state. Value '000' in EV field corresponds to '00000001' in SPT_CS_EVTREG1[EVTREG1] Value '001' in EV field corresponds to '00000010' in SPT_CS_EVTREG1[EVTREG1] Value '010' in EV field corresponds to '00000100' in SPT_CS_EVTREG1[EVTREG1] Value '011' in EV field corresponds to '00001000' in SPT_CS_EVTREG1[EVTREG1] Value '100' in EV field corresponds to '00010000' in SPT_CS_EVTREG1[EVTREG1] Value '101' in EV field corresponds to '00100000' in SPT_CS_EVTREG1[EVTREG1] Value '110' in EV field corresponds to '01000000' in SPT_CS_EVTREG1[EVTREG1] Value '111' in EV field corresponds to '10000000' in SPT_CS_EVTREG1[EVTREG1]

45.8.66 EVT2 Status Register (SPT_CS_EVTREG2)

Address: 0h base + 130h offset = 130h

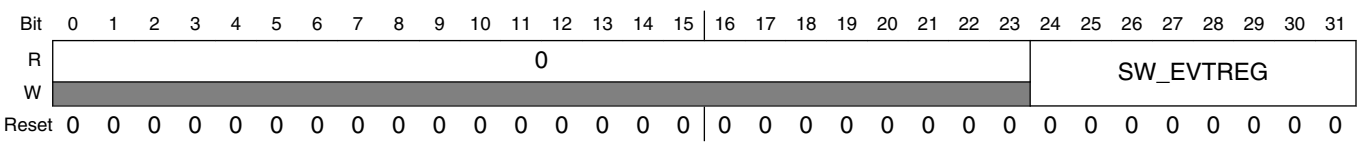


SPT_CS_EVTREG2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 EVTREG2	EVT2 register Each bit corresponds to an event. Whenever an EVT command occurs a bit corresponding to the specified event (given in the command word by the field name 'EV') is set when the field 'EV_LVL' in the command word is 1 or reset when 'EV_LVL' is 0. This field is internally reset in the SETUP state. Value '000' in EV field corresponds to '00000001' in SPT_CS_EVTREG2[EVTREG2] Value '001' in EV field corresponds to '00000010' in SPT_CS_EVTREG2[EVTREG2] Value '010' in EV field corresponds to '00000100' in SPT_CS_EVTREG2[EVTREG2] Value '011' in EV field corresponds to '00001000' in SPT_CS_EVTREG2[EVTREG2] Value '100' in EV field corresponds to '00010000' in SPT_CS_EVTREG2[EVTREG2] Value '101' in EV field corresponds to '00100000' in SPT_CS_EVTREG2[EVTREG2] Value '110' in EV field corresponds to '01000000' in SPT_CS_EVTREG2[EVTREG2] Value '111' in EV field corresponds to '10000000' in SPT_CS_EVTREG2[EVTREG2]

45.8.67 SW Event Trigger Register (SPT_CS_SW_EVTREG)

Address: 0h base + 134h offset = 134h

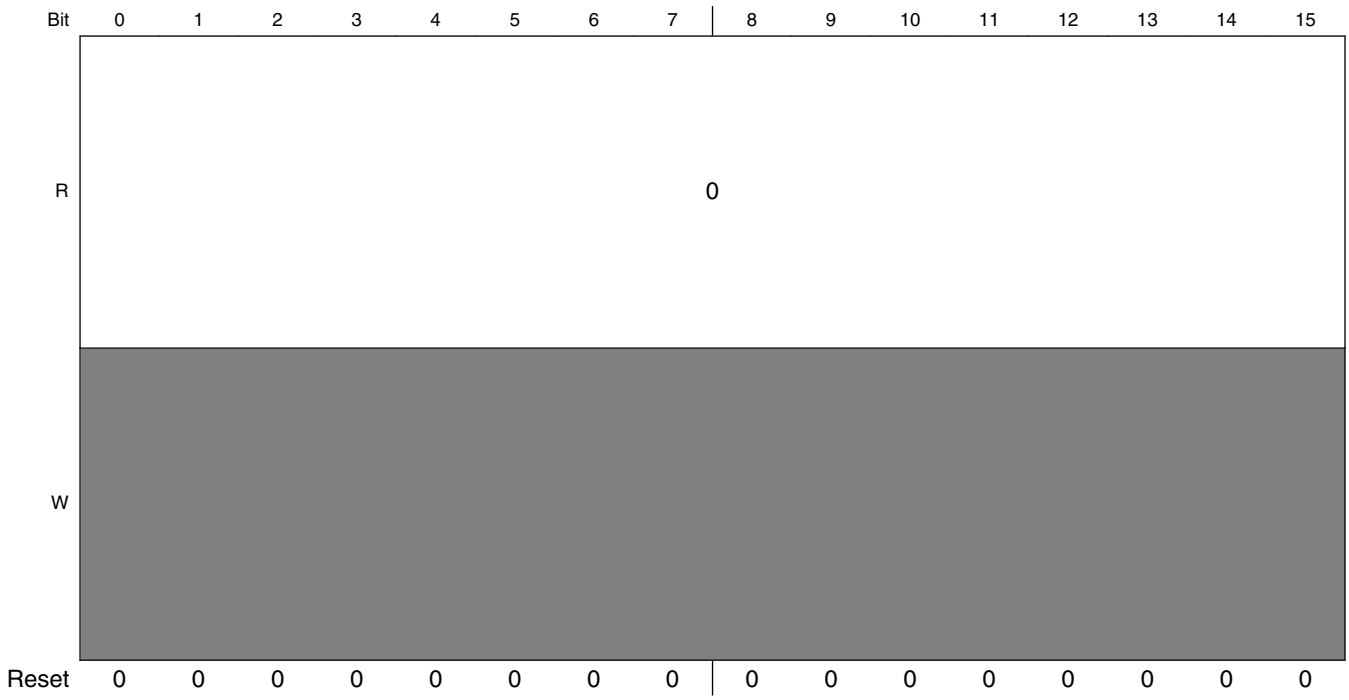


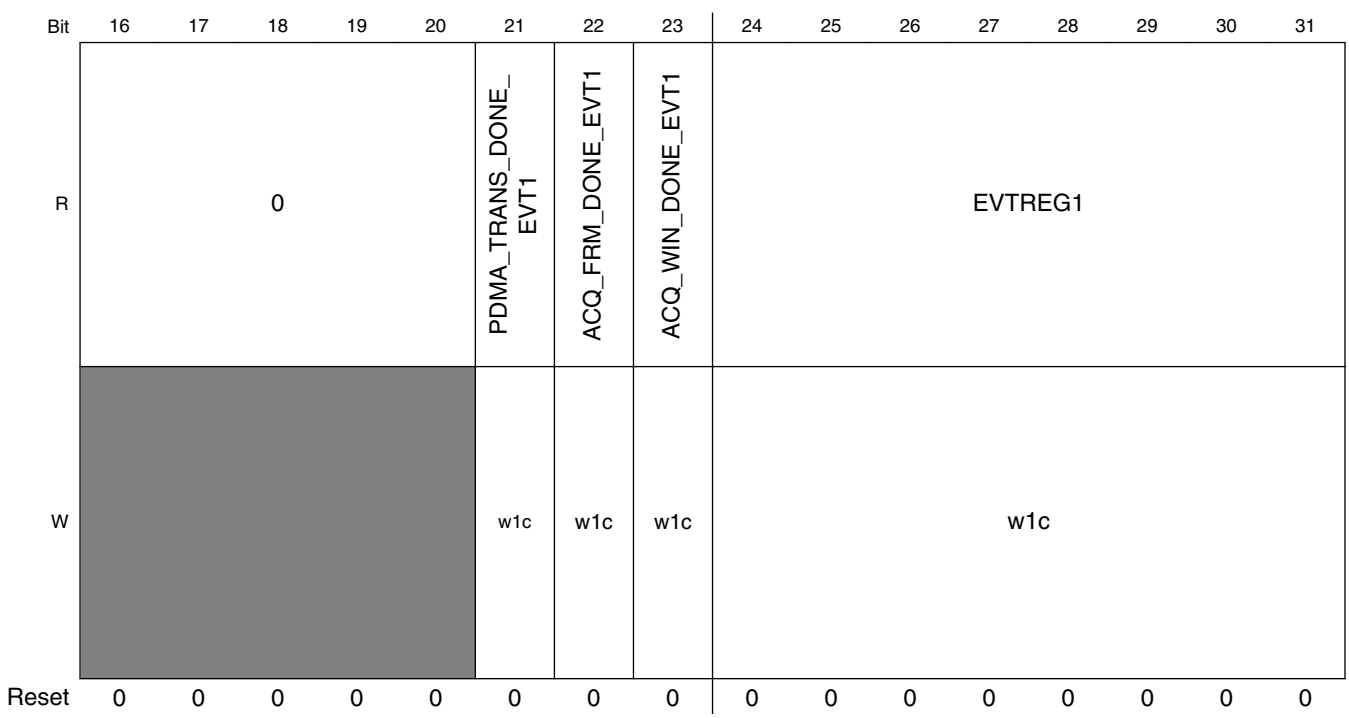
SPT_CS_SW_EVTREG field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SW_EVTREG	SW event trigger SW can trigger an event on any of the 8 bits, to be used by WAIT command. After WAIT on these events is over they are reset internally., and hence these events can only be positive-edge triggered or positive-level triggered They are also internally reset in the SETUP state

45.8.68 Core 1 Version Register Events (SPT_CORE1_VER_EVT)

Address: 0h base + 138h offset = 138h





SPT_CORE1_VER_EVT field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PDMA_TRANS_ DONE_EVT1	Event for PDMA transfer done for CPU core 1 This field is internally reset in the SETUP state.
22 ACQ_FRM_ DONE_EVT1	Event for acquisition frame done for CPU core 1 This field is internally reset in the SETUP state.
23 ACQ_WIN_ DONE_EVT1	Event for acquisition window finished for CPU core 1 This field is internally reset in the SETUP state.
24–31 EVTREG1	EVT1 register Each bit corresponds to an event. Whenever an EVT command occurs a bit corresponding to the specified event (given in the command word by the field name 'EV') is set when the field 'EV_LVL' in the command word is 1 or reset when 'EV_LVL' is 0. This field is internally reset in the SETUP state. Value '000' in EV field corresponds to '00000001' in SPT_CS_EVTREG1[EVTREG1] Value '001' in EV field corresponds to '00000010' in SPT_CS_EVTREG1[EVTREG1] Value '010' in EV field corresponds to '00000100' in SPT_CS_EVTREG1[EVTREG1] Value '011' in EV field corresponds to '00001000' in SPT_CS_EVTREG1[EVTREG1] Value '100' in EV field corresponds to '00010000' in SPT_CS_EVTREG1[EVTREG1] Value '101' in EV field corresponds to '00100000' in SPT_CS_EVTREG1[EVTREG1] Value '110' in EV field corresponds to '01000000' in SPT_CS_EVTREG1[EVTREG1] Value '111' in EV field corresponds to '10000000' in SPT_CS_EVTREG1[EVTREG1].

45.8.69 Core 2 Version Register Events (SPT_CORE2_VER_EVT)

Address: 0h base + 13Ch offset = 13Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

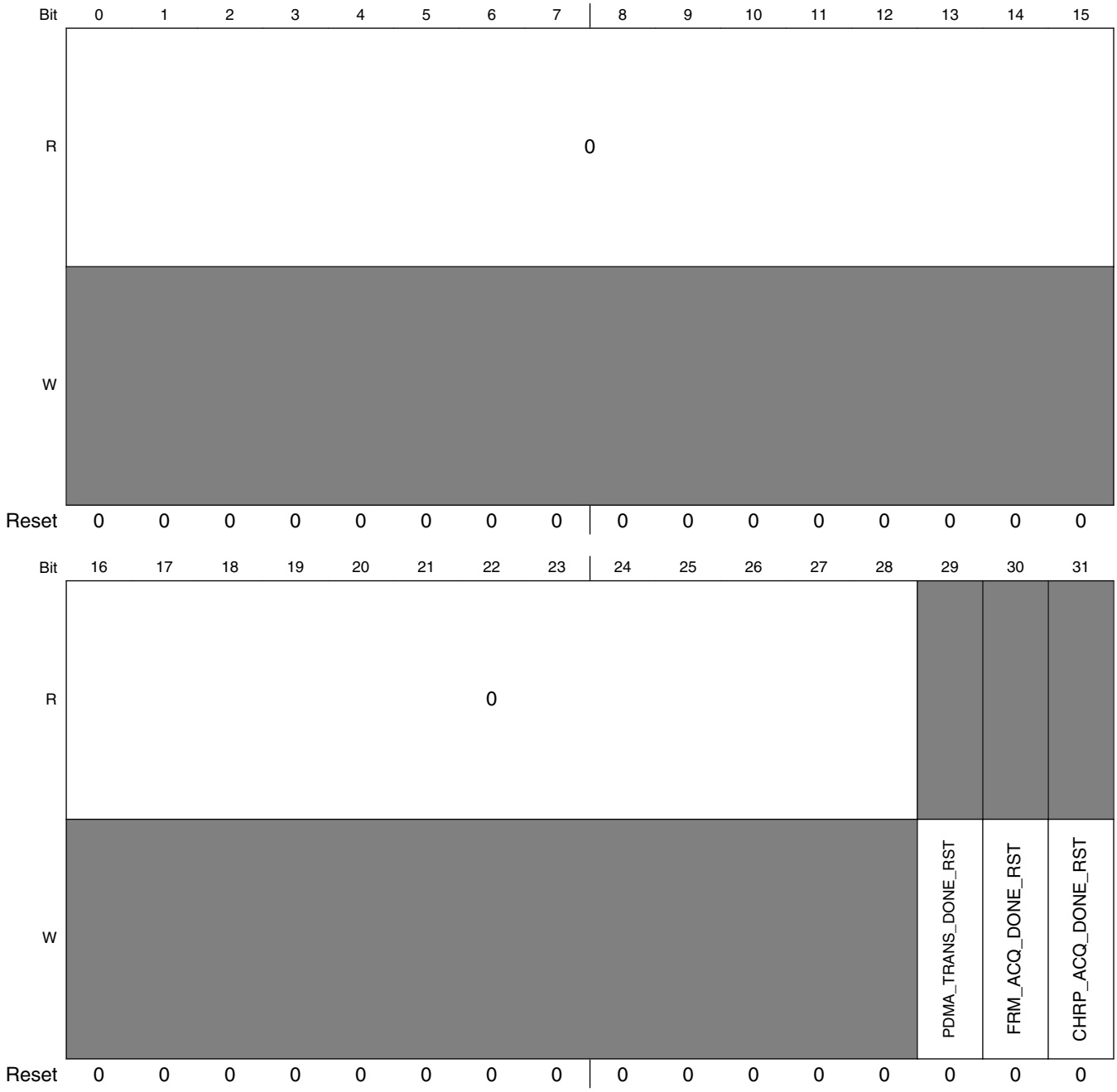
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PDMA_TRANS_DONE_EVT2	ACQ_FRM_DONE_EVT2	ACQ_WIN_DONE_EVT2	EVTREG2							
W						w1c	w1c	w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CORE2_VER_EVT field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PDMA_TRANS_ DONE_EVT2	Event for PDMA transfer done for CPU core 2 This field is internally reset in the SETUP state.
22 ACQ_FRM_ DONE_EVT2	Event for acquisition frame done for CPU core 2 This field is internally reset in the SETUP state.
23 ACQ_WIN_ DONE_EVT2	Event for acquisition window finished for CPU core 2 This field is internally reset in the SETUP state.
24–31 EVTREG2	EVT2 register Each bit corresponds to an event. Whenever an EVT command occurs a bit corresponding to the specified event (given in the command word by the field name 'EV') is set when the field 'EV_LVL' in the command word is 1 or reset when 'EV_LVL' is 0. This field is internally reset in the SETUP state. Value '000' in EV field corresponds to '00000001' in SPT_CS_EVTREG2[EVTREG2] Value '001' in EV field corresponds to '00000010' in SPT_CS_EVTREG2[EVTREG2] Value '010' in EV field corresponds to '00000100' in SPT_CS_EVTREG2[EVTREG2] Value '011' in EV field corresponds to '00001000' in SPT_CS_EVTREG2[EVTREG2] Value '100' in EV field corresponds to '00010000' in SPT_CS_EVTREG2[EVTREG2] Value '101' in EV field corresponds to '00100000' in SPT_CS_EVTREG2[EVTREG2] Value '110' in EV field corresponds to '01000000' in SPT_CS_EVTREG2[EVTREG2] Value '111' in EV field corresponds to '10000000' in SPT_CS_EVTREG2[EVTREG2].

45.8.70 SPT Event Reset Control Register (SPT_EVENT_RST_CTRL)

Address: 0h base + 140h offset = 140h



SPT_EVENT_RST_CTRL field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SPT_EVENT_RST_CTRL field descriptions (continued)

Field	Description
29 PDMA_TRANS_ DONE_RST	When a 1 is written to this bit it causes the registered PDMA Transfer Done event inside the WAIT module to be reset. This bit auto-resets the next cycle.
30 FRM_ACQ_ DONE_RST	When a 1 is written to this bit it causes the registered Frame Acquisition Done event inside the WAIT module to be reset. This bit auto-resets the next cycle.
31 CHRP_ACQ_ DONE_RST	When a 1 is written to this bit it causes the registered Chirp Acquisition Done event inside the WAIT module to be reset. This bit auto-resets the next cycle.

45.8.71 LFSR Load High Value (SPT_PDMA_LFSR_LOAD_VAL_HIGH)

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PDMA_LFSR_LOAD_VAL_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_PDMA_LFSR_LOAD_VAL_HIGH field descriptions

Field	Description
0–31 PDMA_LFSR_ LOAD_VAL_ HIGH	LFSR High value [63:32] for compression

45.8.72 LFSR Load Low Value (SPT_PDMA_LFSR_LOAD_VAL_LOW)

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PDMA_LFSR_LOAD_VAL_LOW																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_PDMA_LFSR_LOAD_VAL_LOW field descriptions

Field	Description
0–31 PDMA_LFSR_ LOAD_VAL_ LOW	LFSR Low Value [31:0] for compression

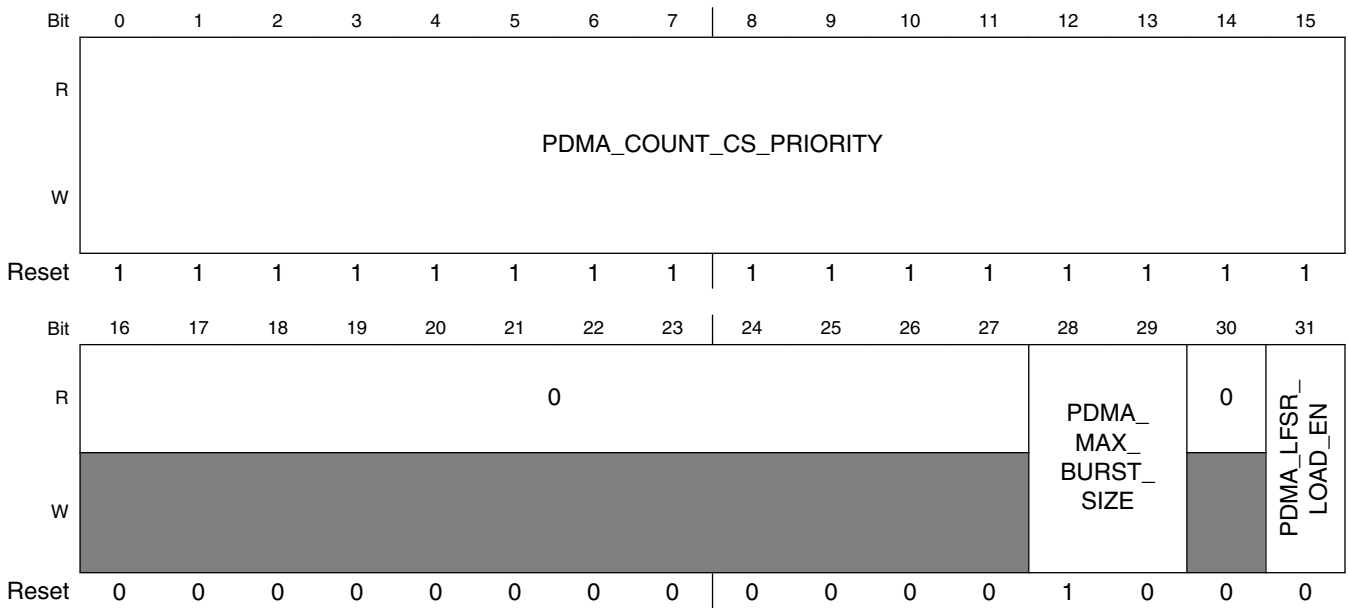
45.8.73 PDMA Control Register (SPT_PDMA_CONTROL)

This register controls the PDMA instruction processing.

NOTE

The minimum value of 'PDMA_COUNT_CS_PRIORITY' field is 0x10. Writing less than 0x10 to PDMA_COUNT_CS_PRIORITY field take value 0x10.

Address: 0h base + 188h offset = 188h



SPT_PDMA_CONTROL field descriptions

Field	Description
0–15 PDMA_COUNT_CS_PRIORITY	PDMA Count register for Command sequencer to get priority Count value in terms of clock cycles that should be loaded after which the command sequencer will get access on the AHB bus
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 PDMA_MAX_BURST_SIZE	Maximum AHB burst size Controls the size of burst that is initiated on the AHB side. 00b - INCR4,01b - INCR8, 10b - INCR16
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDMA_LFSR_LOAD_EN	LFSR load enable bit Enables loading LFSR with register values

45.8.74 PDMA Transfer Count Status (SPT_PDMA_TRANSFER_COUNT_STATUS)

Address: 0h base + 1C0h offset = 1C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AGGR_DATA_COUNT																TRANSFER_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_PDMA_TRANSFER_COUNT_STATUS field descriptions

Field	Description
0–15 AGGR_DATA_COUNT	Aggregation Data Count Specifies the number of aggregated data after performing aggregation operation.
16–31 TRANSFER_COUNT	Number of words transferred via PDMA Number of words transferred in current cycle. This is automatically cleared on the next instruction start.

45.8.75 PDMA formatB Exponent Address status Register (SPT_PDMA_FMTB_EXP_ADDR_STATUS)

Address: 0h base + 1C4h offset = 1C4h

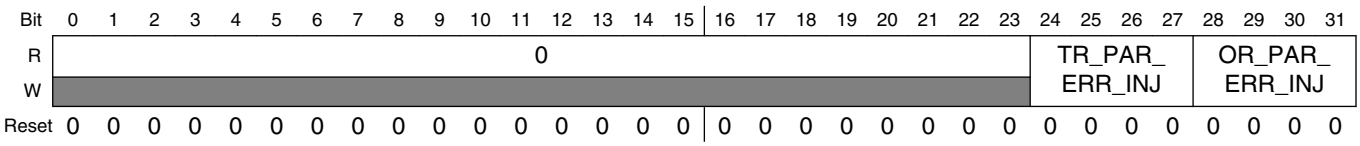
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EXPN_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_PDMA_FMTB_EXP_ADDR_STATUS field descriptions

Field	Description
0–31 EXPN_ADDR	Exponent Address First exponent Address for FormatB Compression and Decompression modes

45.8.76 Memory Error Injection Register (SPT_MEM_ERR_INJECT_CTRL)

Address: 0h base + 1FCh offset = 1FCh

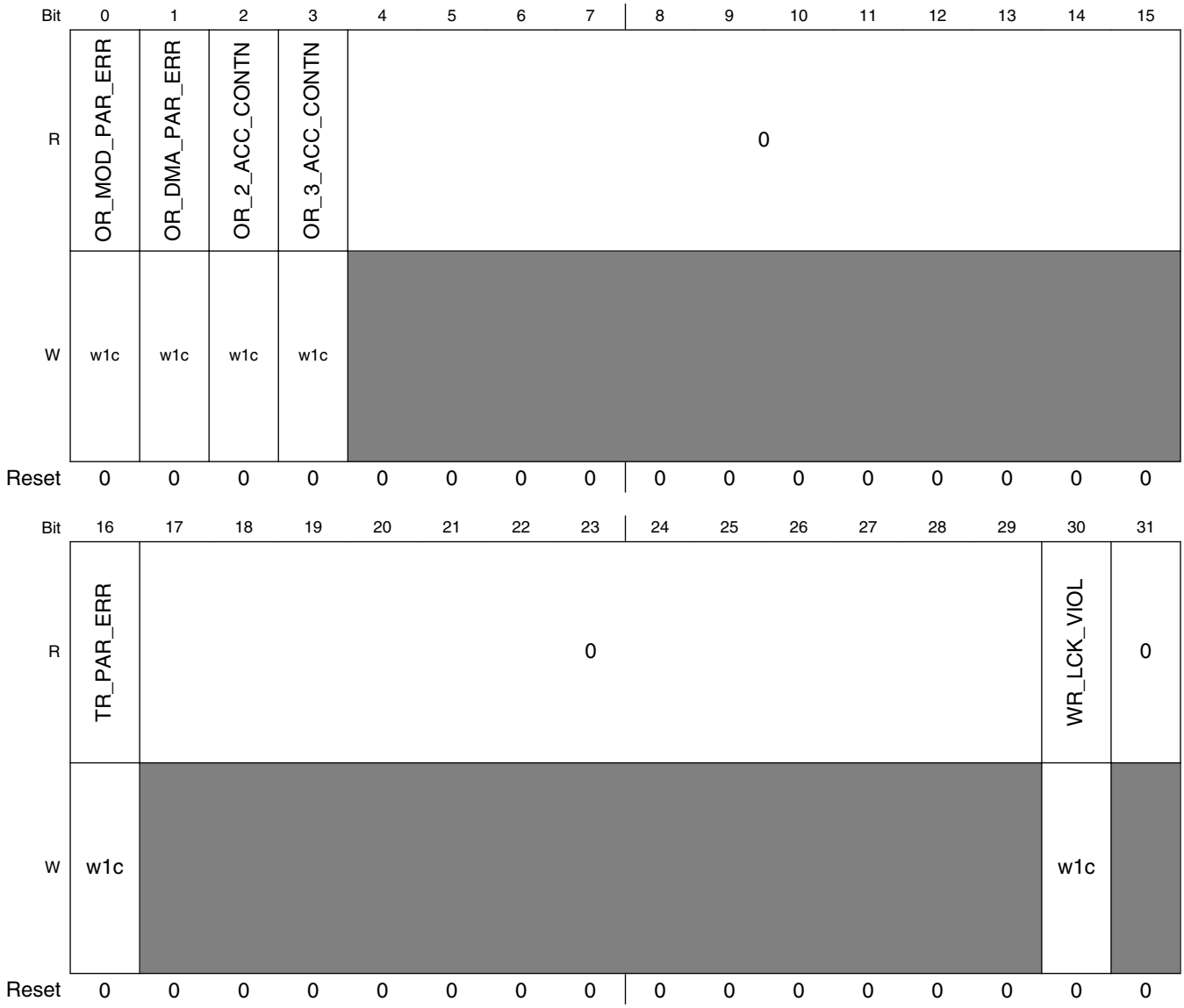


SPT_MEM_ERR_INJECT_CTRL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 TR_PAR_ERR_ INJ	Twiddle RAM parity error Injection This field can be used to inject error in the parity bits in the TRAM. There are 4 bits (1 bit for each parity bit) to invert the actual parity during the data write to the memory. Anyone or all the bits can be set to inject parity error in the corresponding parity bit.
28–31 OR_PAR_ERR_ INJ	Operand RAM parity error Injection This field can be used to inject error in the parity bits in the ORAM. There are 4 bits (1 bit for each parity bit) to invert the actual parity during the data write to the memory. Anyone or all the bits can be set to inject parity error in the corresponding parity bit.

45.8.77 Memory Error Status Register (SPT_MEM_ERR_STATUS)

Address: 0h base + 200h offset = 200h



SPT_MEM_ERR_STATUS field descriptions

Field	Description
0 OR_MOD_PAR_ERR	Operand RAM module parity error Set when read data requested by any of the modules (hardware accelerators) has a parity error
1 OR_DMA_PAR_ERR	Operand RAM DMA parity error Set when read data requested by PDMA has a parity error

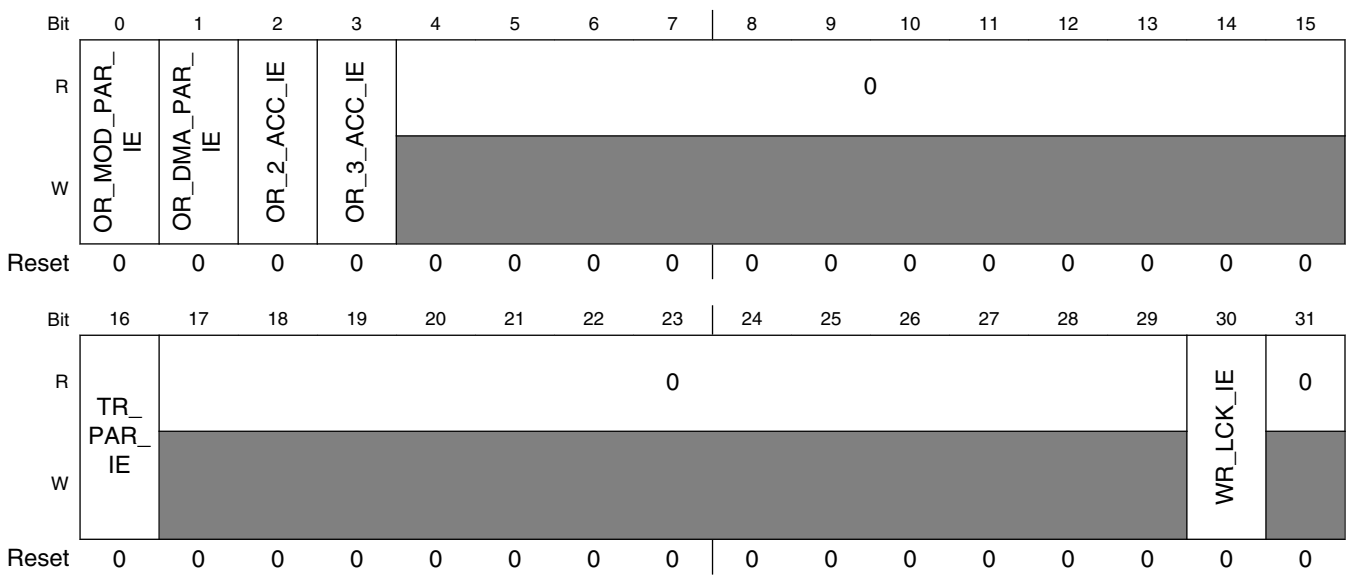
Table continues on the next page...

SPT_MEM_ERR_STATUS field descriptions (continued)

Field	Description
2 OR_2_ACC_CONTN	Operand RAM two access contention Set when two accesses are made on the same operand RAM bank.
3 OR_3_ACC_CONTN	Operand RAM three access contention Set when three accesses are made on the same operand RAM bank.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TR_PAR_ERR	Twiddle RAM parity error Set when any read access has a parity error
17–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 WR_LCK_VIOL	Work register lock violation Set when SPT writes to a work register locked for CPU
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.78 Memory Interrupt Enable register (SPT_MEM_ERR_INT_EN)

Address: 0h base + 204h offset = 204h



SPT_MEM_ERR_INT_EN field descriptions

Field	Description
0 OR_MOD_PAR_IE	Operand RAM module parity interrupt enable

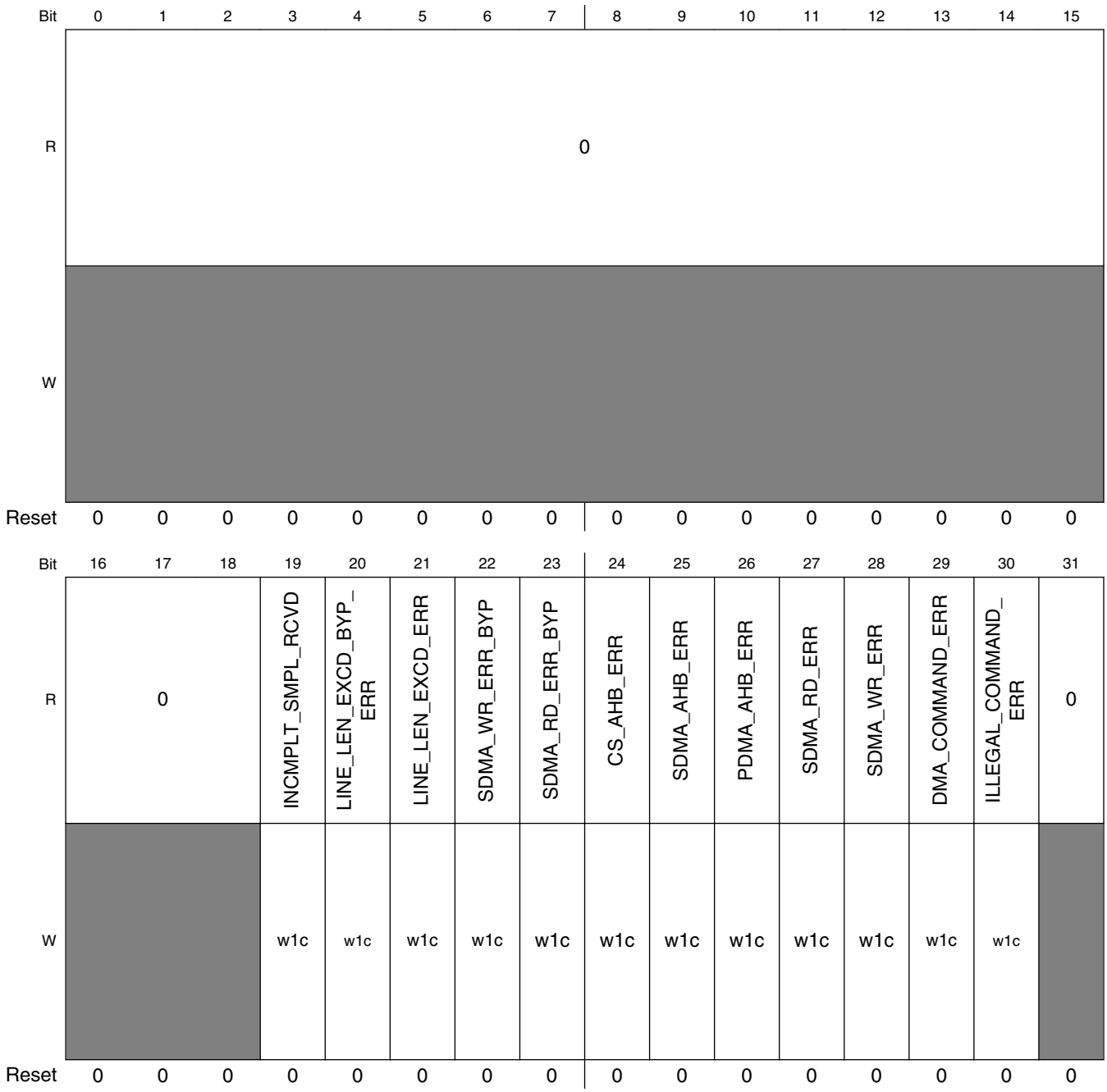
Table continues on the next page...

SPT_MEM_ERR_INT_EN field descriptions (continued)

Field	Description
	Enables the interrupt for MEM_ERR_REG[OP_MOD_PAR_ERR]. Interrupt generated when OP_MOD_PAR_ERR is set.
1 OR_DMA_PAR_IE	Operand RAM DMA parity interrupt enable Enables the interrupt for MEM_ERR_REG[OP_DMA_PAR_ERR]. Interrupt generated when OP_DMA_PAR_ERR is set.
2 OR_2_ACC_IE	Operand RAM two access contention interrupt enable Enables the interrupt for MEM_ERR_REG[OP_2_ACC_CONTN]. Interrupt generated when OP_2_ACC_CONTN is set.
3 OR_3_ACC_IE	Operand RAM three access contention interrupt enable Enables the interrupt for MEM_ERR_REG[OP_3_ACC_CONTN]. Interrupt generated when OP_3_ACC_CONTN is set.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TR_PAR_IE	Twiddle RAM parity error interrupt enable Enables the interrupt for MEM_ERR_REG[TR_PAR_ERR]. Interrupt generated when TR_PAR_ERR is set.
17–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 WR_LCK_IE	Work register lock interrupt enable Enables the interrupt for MEM_ERR_REG[WR_LCK_VIOL]. Interrupt generated when WR_LCK_VIOL is set.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.79 DMA Error Status Register (SPT_DMA_ERR_STATUS)

Address: 0h base + 208h offset = 208h



SPT_DMA_ERR_STATUS field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SPT_DMA_ERR_STATUS field descriptions (continued)

Field	Description
19 INCMPLT_ SMPL_RCVD	Incomplete sample received An exception is generated when the acquisition window deasserts before the number of samples programmed for regular SDMA or bypass SDMA is achieved. The acquisition block will continue working and no interrupt is generated corresponding to this exception.
20 LINE_LEN_ EXCD_BYP_ ERR	Bypass SDMA line length exceeded This error is asserted when cfg_trf_mode is high, that is 2D mode in acquisition, if the incrementing address exceeds the base address of next chirp. **NOTE:** W1C should be performed between the subsequent chirp start pulse and the acquisition window start to avoid setting of the error bit again. Also, it is expected that the line length is reprogrammed if such an error occurs.
21 LINE_LEN_ EXCD_ERR	Line Length Exceeded This error is asserted when cfg_trf_mode is high, that is 2D mode in acquisition, if the incrementing address exceeds the base address of next chirp. **NOTE:** W1C should be performed between the subsequent chirp start pulse and the acquisition window start to avoid setting of the error bit again. Also, it is expected that the line length is reprogrammed if such an error occurs.
22 SDMA_WR_ ERR_BYP	Bypass SDMA write error Generated if the Bypass SDMA internal buffer was full (due to arbiter being busy) and Acquisition data write was performed.
23 SDMA_RD_ ERR_BYP	Bypass SDMA read error Generated if the Bypass SDMA internal buffer was empty and arbiter tries to perform SRAM write via SDMA.
24 CS_AHB_ERR	Command Sequencer AHB error Response Generated if there is an AHB error response for Command Sequencer access. In case of access to a protected region of memory it is possible that this error indication may not be given, but an error status will be given on SPT_CS_STATUS1[ILL_OPCODE] status bit and corresponding interrupt can be enabled in SPT_CS_INTEN1. It is also recommended that the user check the slave memory controller error indication.
25 SDMA_AHB_ ERR	SDMA AHB Error Response Generated if there is an AHB error response for SDMA access
26 PDMA_AHB_ ERR	PDMA AHB Error Generated if there is an AHB error response for PDMA access
27 SDMA_RD_ERR	SDMA Read Error Generated if the SDMA internal buffer was empty and arbiter tries to perform SRAM write via SDMA.
28 SDMA_WR_ERR	SDMA Write Error Generated if the SDMA internal buffer was full (due to arbiter being busy) and Acquisition data write was performed.

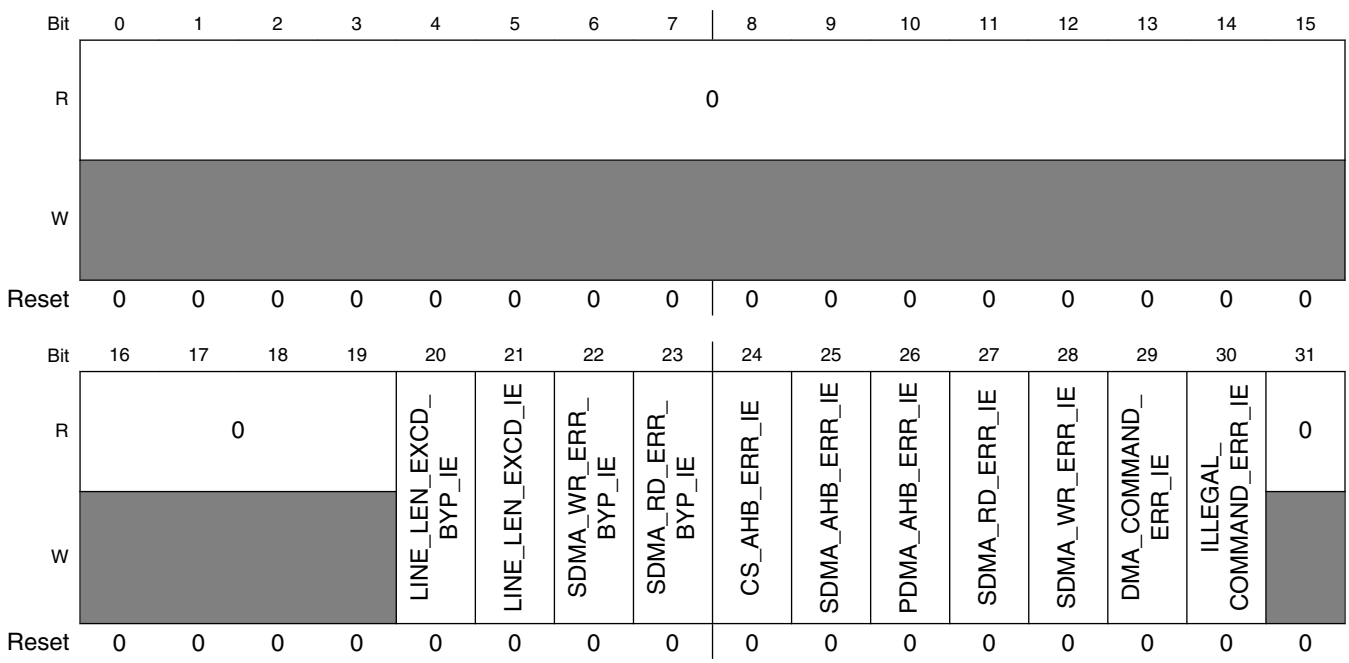
Table continues on the next page...

SPT_DMA_ERR_STATUS field descriptions (continued)

Field	Description
29 DMA_ COMMAND_ERR	Generated if the transfer size is more than the maximum value possible
30 ILLEGAL_ COMMAND_ ERR	Generated if proprietary functionality is tried being used by user.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.8.80 Interrupt Enable for DMA_ERROR_STATUS (SPT_DMA_ERR_INT_EN)

Address: 0h base + 20Ch offset = 20Ch



SPT_DMA_ERR_INT_EN field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 LINE_LEN_ EXCD_BYP_IE	Bypass SDMA line length exceeded interrupt Enables or disables interrupt generation 0 Interrupt generation disabled 1 Interrupt generation enabled

Table continues on the next page...

SPT_DMA_ERR_INT_EN field descriptions (continued)

Field	Description
21 LINE_LEN_ EXCD_IE	Line length exceeded interrupt Enables or disables interrupt generation 0 Interrupt generation disabled 1 Interrupt generation enabled
22 SDMA_WR_ ERR_BYP_IE	Bypass SDMA write error interrupt Enables or disables interrupt generation 0 Interrupt generation disabled 1 Interrupt generation enabled
23 SDMA_RD_ ERR_BYP_IE	Bypass SDMA read error interrupt Enables or disables interrupt generation. 0 Interrupt generation disabled 1 Interrupt generation enabled
24 CS_AHB_ERR_ IE	Command Sequencer AHB Interrupt Enable Enables or disables interrupt generation.
25 SDMA_AHB_ ERR_IE	SDMA AHB Error Interrupt Enable Enables or disables interrupt generation.
26 PDMA_AHB_ ERR_IE	Interrupt Enable for PDMA AHB Error Enables or disables interrupt generation.
27 SDMA_RD_ ERR_IE	SDMA Read Error Interrupt Enable Enables or disables interrupt generation. 0 Interrupt generation disabled. 1 Interrupt generation enabled.
28 SDMA_WR_ ERR_IE	SDMA Write Error Interrupt Enable Enables or disables interrupt generation. 0 Interrupt generation disabled. 1 Interrupt generation enabled.
29 DMA_ COMMAND_ ERR_IE	Generated if the transfer size is more than the maximum value possible
30 ILLEGAL_ COMMAND_ ERR_IE	Generated if proprietary functionality is tried being used by user.
31 Reserved	Interrupt Enable bit This field is reserved. This read-only field is reserved and always has the value 0.

45.8.81 Global Status Register (SPT_GBL_STATUS)

GBL_STATUS is a w1c register but should only be written to as a full register, never as single bits. This is because a single bit write is compiled into a read-modify-write instruction sequence that clears the other bits simultaneously. Clearing by writing the full register with a mask avoids this situation. All the bits in this register are reset in SETUP state.

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													PDMA_TRANS_DONE	FRM_ACQ_DONE	CHRP_ACQ_DONE
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_GBL_STATUS field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PDMA_TRANS_DONE	PDMA transfer done
30 FRM_ACQ_DONE	Frame Acquisition Done This status field is set when a Frame has completed and acquisition of all chirps within this frame are done and have been transferred to System RAM. All statistics for this frame are stable and can be read. Refer to explanation added in interrupts section. This status will never set if ACQ_GBL_CTRL_0[INF_CHRPS] is set. 0 Frame Acquisition is not done 1 Frame Acquisition is done
31 CHRP_ACQ_DONE	Chirp Acquisition Done This status field is set when a chirp has completed and acquisition of all samples within this chirp are done and have been transferred to System RAM. All statistics for this chirp are now stable and can be read. An internal chirp counter is used to detect the last chirp in a frame based on the programmed value of ACQ_GBL_CTRL_1[NUM_CHRP]. This counter decrements every time acquisition window within a given chirp de-asserts, thus signifying end of acquisition for that chirp. 0 Chirp Acquisition is not done 1 Chirp acquisition is done

45.8.82 Global Status Interrupt Enable Register (SPT_GBL_STATUS_IE)

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													PDMA_TRANS_DONE_IE	FRM_ACQ_DONE_IE	CHRP_ACQ_DONE_IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

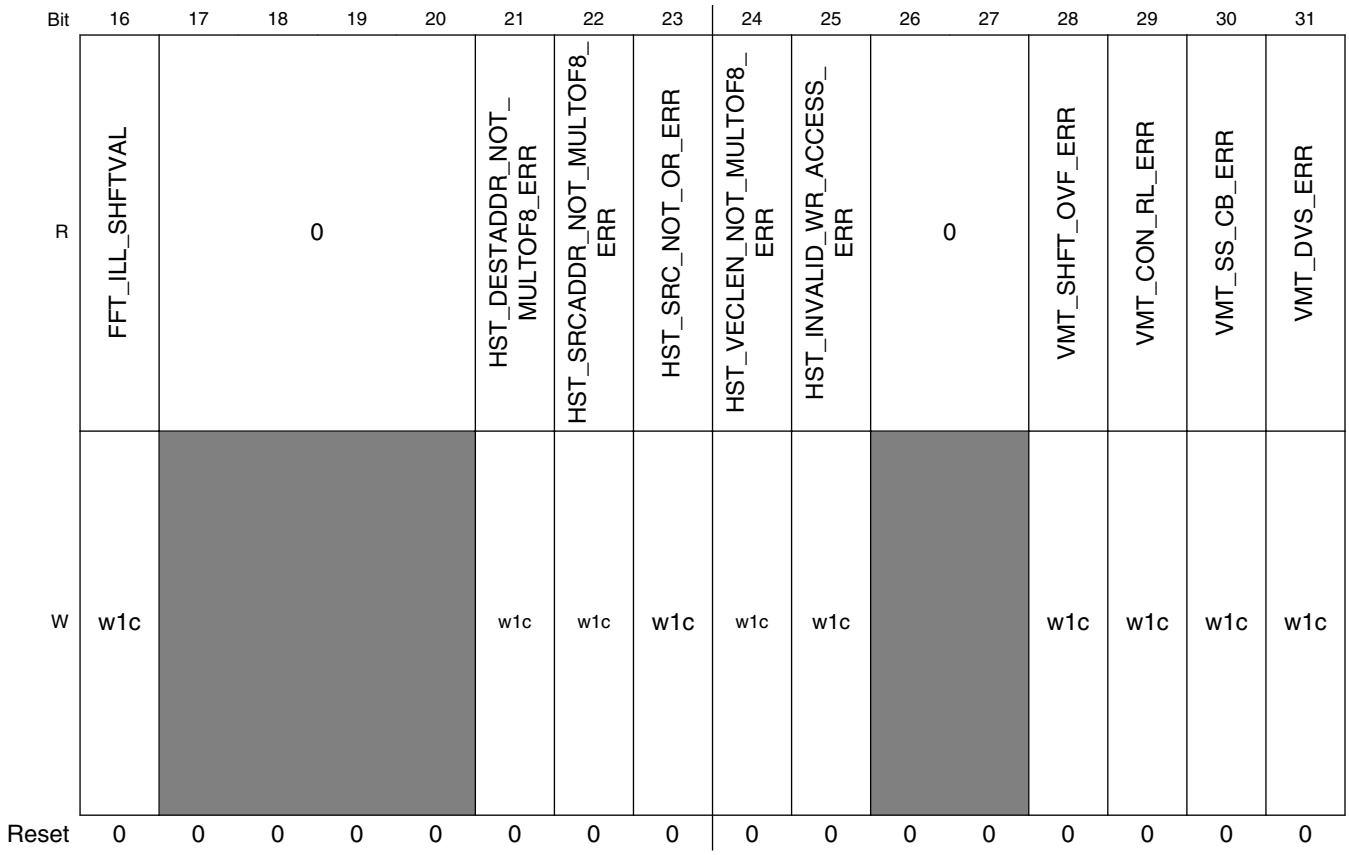
SPT_GBL_STATUS_IE field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PDMA_TRANS_ DONE_IE	PDMA transfer done interrupt enable
30 FRM_ACQ_ DONE_IE	Frame Acquisition Done Interrupt Enable Set this field to enable the interrupt. 0 Interrupt Disabled 1 Interrupt Enabled
31 CHRP_ACQ_ DONE_IE	Chirp Acquisition Done Interrupt Enable Set this field to enable the interrupt 0 Interrupt Disabled 1 Interrupt Enabled

45.8.83 Hardware Accelerator Error Status Register (SPT_HW_ACC_ERR_STATUS)

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FFT_RDX2_RND_ERR	FFT_OPR_ADDR_ERR	FFT_MULT_COEF2_ERR	FFT_MULT_COEF1_ERR	FFT_TW_OVS_ERR	FFT_QE_VL_OVS_ERR	FFT_RDX4_RND_ERR	FFT_WIN_RND_ERR	MAXS_IP_CMD_ERR	0			COPY_IP_CMD_ERR	0		
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FFT_ILL_SHFTVAL	0				HST_DESTADDR_NOT_MULTOF8_ERR	HST_SRCADDR_NOT_MULTOF8_ERR	HST_SRC_NOT_OR_ERR	HST_VECLEN_NOT_MULTOF8_ERR	HST_INVALID_WR_ACCESS_ERR	0		VMT_SHFT_OVF_ERR	VMT_CON_RL_ERR	VMT_SS_CB_ERR	VMT_DVS_ERR
W	w1c					w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_HW_ACC_ERR_STATUS field descriptions

Field	Description
0 FFT_RDX2_RND_ERR	FFT Radix2 Round Error This bit is set if an invalid value of RDX2 round is provided for a given vector length
1 FFT_OPR_ADDR_ERR	Operand Address Error This bit indicates source or destination address for FFT Core Operands are not from OPRAM
2 FFT_MULT_COEF2_ERR	FFT MULT Co-efficient 2 Error This field is set if MULT_COEF_ADDR is ORAM for Windowing operation with MULT_MODE = 1.
3 FFT_MULT_COEF1_ERR	FFT MULT Co-efficient 1 Error This field is set if MULT_COEF_ADDR / TAP_COEF_ADDR is not TRAM for the following: a) Non-Windowing operation b) Windowing operation with MULT_MODE = 2
4 FFT_TW_OVS_ERR	FFT Twiddle Oversampling Error This bit is set when an invalid twiddle oversampling option is selected. The invalid scenarios are: Vector length = 32 and twiddle oversampling factor field value > 7 Vector length = 64 and twiddle oversampling factor field value > 6 Vector length = 128 and twiddle oversampling factor field value > 5 Vector length = 256 and twiddle oversampling factor field value > 4 Vector length = 512 and twiddle oversampling factor field value > 3 Vector length = 1024 and twiddle oversampling factor field value > 2 Vector length = 2048 and twiddle oversampling factor field value > 1 Vector length = 4096 and twiddle oversampling factor field value > 0

Table continues on the next page...

SPT_HW_ACC_ERR_STATUS field descriptions (continued)

Field	Description
5 FFT_QE_VL_ OVS_ERR	Quadrature Extension Vector Length Oversampling Error This bit is set if QE is not enabled for the vector length and Twiddle oversampling field combinations which require QE to be enabled. This bit is set if : Vector length = 16 and twiddle oversampling factor field value > 5 Vector length = 32 and twiddle oversampling factor field value > 4 Vector length = 64 and twiddle oversampling factor field value > 3 Vector length = 128 and twiddle oversampling factor field value > 2 Vector length = 256 and twiddle oversampling factor field value > 1 Vector length = 512 and twiddle oversampling factor field value > 0
6 FFT_RDX4_ RND_ERR	FFT Radix4 Round Error This bit is set if an invalid value of RDX4 round is provided for a given vector length
7 FFT_WIN_RND_ ERR	FFT Window Round Error This bit indicates if Round is greater than 0 in Radix4 Command with WIN bit high.
8 MAXS_IP_CMD_ ERR	MAXS input command error Asserted when there is an error in the incoming instruction 0 No command error 1 Asserted when any of the following errors are encountered in the incoming instruction. Refer Error scenarios
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 COPY_IP_CMD_ ERR	COPY input command error Asserted when there is an error in the incoming instruction 0 No command error 1 Asserted when any of the following command errors are encountered: 1.In in_dattyp = CMPLX and preproc = 0 (no preproc) 2.If in_dattyp != CMPLX and preproc = 2 (MAG) 3.If (src_add[15:14] = WREG_ADD or dest_add[15:14] = WREG_ADD) 4.In_dattyp == 11b (RESERVED) 5.Preproc == 11b (RESERVED) 6.Cp_type >= 1100b 7.Cp_type == CP_TRNSPOSE and vec_sz[5:0] != 0 8.Vec_sz[3:0] != 0 9.Cp_type == CP_SHFT AND dest_add[3:0] == 0 // copy shift with 0 shift not allowed
13–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FFT_ILL_ SHFTVAL	FFT Illegal Shift Value error This bit is raised when the mult_mode=0 and pre-shift value comes from WR.
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 HST_ DESTADDR_ NOT_MULTOF8_ ERR	Destination address not multiple of 8 error It is set when the destination address value for HIST instruction is not a multiple of 8. This condition is checked only when destination is either operand RAM or twiddle RAM. This field is internally reset in SETUP state.
22 HST_ SRCADDR_ NOT_MULTOF8_ ERR	Source address not multiple of 8 error It is set when the source address value for HIST instruction is not a multiple of 8. This condition is not checked for 'write only' operation mode. This field is internally reset in SETUP state.

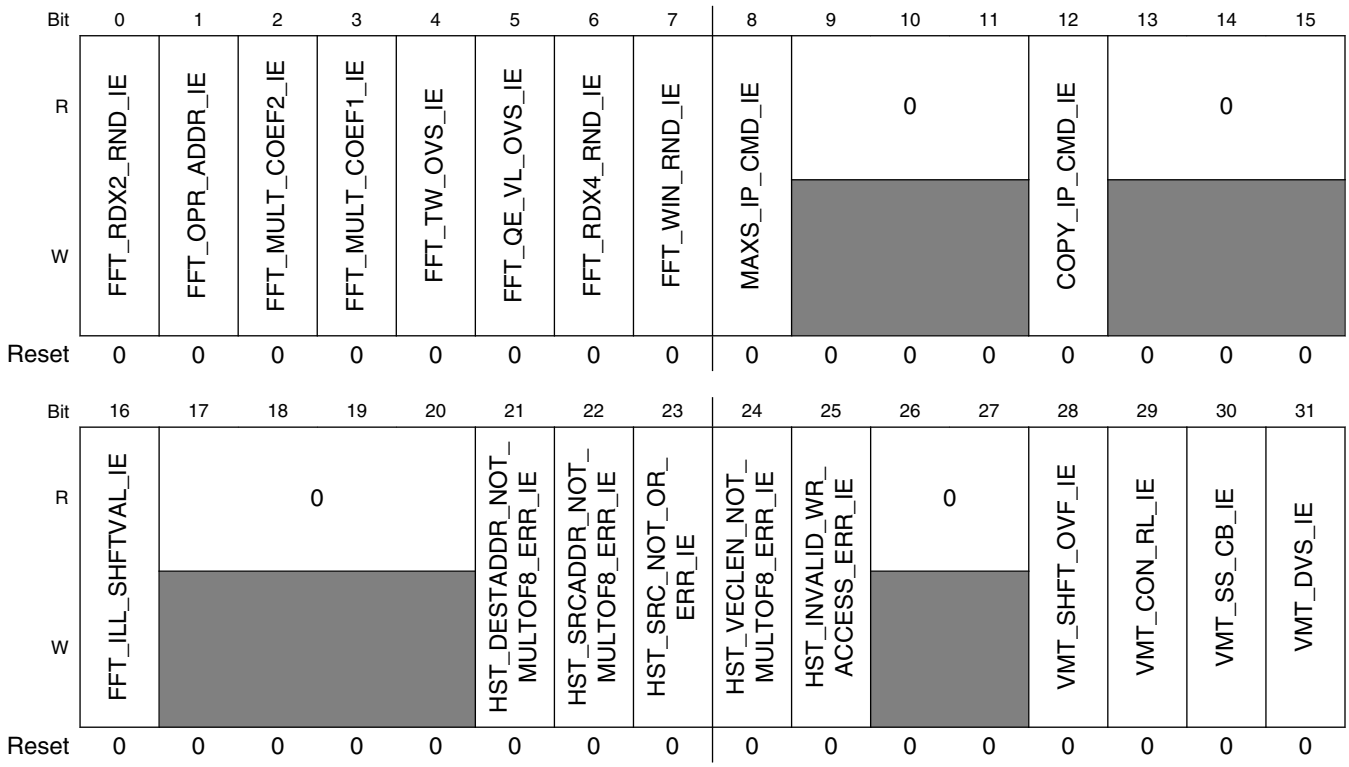
Table continues on the next page...

SPT_HW_ACC_ERR_STATUS field descriptions (continued)

Field	Description
23 HST_SRC_NOT_OR_ERR	Source not Operand RAM error It is set when the source address for the HIST instruction is any other than that of Operand RAM. This condition is not checked for the 'write only' operation mode. This field is internally reset in SETUP state.
24 HST_VECLEN_NOT_MULTOF8_ERR	Vector length not multiple of 8 error It is set when the vector length for HIST instruction is not a multiple of 8. This field is internally reset in SETUP state.
25 HST_INVALID_WR_ACCESS_ERR	HIST work register access error If the IMA bit is set in the HIST instruction and it points from work register number 48 to work register number 63 it is an access error condition. This field is internally reset in SETUP state.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 VMT_SHFT_OVF_ERR	VMT Shift Overflow Error This field is set when in VMT, Vector Shift operation leads to saturation, due to overflow .However, this overflow condition does not cause the command sequencer to go to the ASYNCSTOP state; the sequencer continues in the same state. 0 No saturation condition hit 1 Saturation condition is hit
29 VMT_CON_RL_ERR	VMT Conjugate with Real Type Error This error field is set if in VMT, conjugate operation is selected when Vector type is Real. 0 No Error 1 Error condition is hit
30 VMT_SS_CB_ERR	VMT Sum Scale with Conjugate/Bypass Error This error status field is set if in VMT, sum and scale is selected along with conjugate or bypass with input type as complex. 0 No Error 1 Error condition is hit
31 VMT_DVS_ERR	VMT Dual vector sum error This error status field is set if in Dual Vector Sum operation of VMT, the first vector source (SRC_ADDR) is selected as TRAMor the second vector source (2ND_VEC_SRC) is selected as ORAM. 0 No Error 1 Error condition is hit

45.8.84 Hardware Accelerator Error Interrupt Enable Register (SPT_HW_ACC_ERR_IE)

Address: 0h base + 21Ch offset = 21Ch



SPT_HW_ACC_ERR_IE field descriptions

Field	Description
0 FFT_RDX2_RND_IE	FFT Radix2 Round Error Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_RDX2_RND_ERR] is asserted.
1 FFT_OPR_ADDR_IE	Operand Address Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_OPR_ADDR_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
2 FFT_MULT_COEF2_IE	FFT MULT Co-efficient 2 Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_MULT_COEF2_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
3 FFT_MULT_COEF1_IE	FFT MULT Co-efficient 1 Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_MULT_COEF1_ERR] is asserted.

Table continues on the next page...

SPT_HW_ACC_ERR_IE field descriptions (continued)

Field	Description
	0 Interrupt Disabled 1 Interrupt Enabled
4 FFT_TW_OVS_IE	FFT Twiddle Oversampling Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_TW_OVS_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
5 FFT_QE_VL_OVS_IE	Quadrature Extension Vector Length Oversampling Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_QE_VL_OVS_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
6 FFT_RDX4_RND_IE	FFT Radix4 Round Error Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_RDX4_RND_ERR] is asserted.
7 FFT_WIN_RND_IE	FFT Window Round Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[FFT_WIN_RND_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
8 MAXS_IP_CMD_IE	MAXS input command interrupt enable If set, an interrupt is generated when the HW_ACC_ERR_STATUS[MAXS_IP_CMD_ERR] is asserted
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 COPY_IP_CMD_IE	COPY input command interrupt enable If set, an interrupt is generated when the HW_ACC_ERR_STATUS[COPY_IP_CMD_ERR] is asserted
13–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FFT_ILL_SHFTVAL_IE	FFT Illegal Shift Value Error This is the interrupt enable for HW_ACC_ERR_STATUS[FFT_ILL_SHFTVAL].
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 HST_DESTADDR_NOT_MULTOF8_ERR_IE	Destination address not multiple of 8 error interrupt enable If set then an interrupt is issued if SPT_HW_ACC_ERR_STATUS[HST_DESTADDR_NOT_MULTOF8_ERR] is set
22 HST_SRCADDR_NOT_MULTOF8_ERR_IE	Source address not multiple of 8 error interrupt enable If set then an interrupt is issued if SPT_HW_ACC_ERR_STATUS[HST_SRCADDR_NOT_MULTOF8_ERR] is set

Table continues on the next page...

SPT_HW_ACC_ERR_IE field descriptions (continued)

Field	Description
23 HST_SRC_NOT_OR_ERR_IE	Source not Operand RAM error interrupt enable If set then an interrupt is issued when SPT_HW_ACC_ERR_STATUS[HST_SRC_NOT_OR_ERR] is set
24 HST_VECLEN_NOT_MULTOF8_ERR_IE	Vector length not multiple of 8 error interrupt enable If set then an interrupt is issued if SPT_HW_ACC_ERR_STATUS[HST_VECLEN_NOT_MULTOF8_ERR] is set
25 HST_INVALID_WR_ACCESS_ERR_IE	HIST work register access error interrupt enable If set then an interrupt is raised when SPT_HW_ACC_ERR_STATUS[HST_INVALID_WR_ACCESS_ERR] is set
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 VMT_SHFT_OVF_IE	VMT Shift Overflow Interrupt Enable. Allows interrupt generation when HW_ACC_ERR_STATUS[VMT_SHFT_OVF_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
29 VMT_CON_RL_IE	VMT Conjugate with Real Type Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[VMT_CON_RL_ERR] is asserted. 0 Interrupt Disabled. 1 Interrupt Enabled
30 VMT_SS_CB_IE	VMT Sum Scale with Conjugate/Bypass Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[VMT_SS_CB_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled
31 VMT_DVS_IE	VMT Dual vector sum Interrupt Enable Allows interrupt generation when HW_ACC_ERR_STATUS[VMT_DVS_ERR] is asserted. 0 Interrupt Disabled 1 Interrupt Enabled

45.8.85 HIST Overflow Status0 Register (SPT_HIST_OVF_STATUS0)

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	B31_OVF	B30_OVF	B29_OVF	B28_OVF	B27_OVF	B26_OVF	B25_OVF	B24_OVF	B23_OVF	B22_OVF	B21_OVF	B20_OVF	B19_OVF	B18_OVF	B17_OVF	B16_OVF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B15_OVF	B14_OVF	B13_OVF	B12_OVF	B11_OVF	B10_OVF	B9_OVF	B8_OVF	B7_OVF	B6_OVF	B5_OVF	B4_OVF	B3_OVF	B2_OVF	B1_OVF	B0_OVF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_HIST_OVF_STATUS0 field descriptions

Field	Description
0 B31_OVF	Bin 31 overflow It is set if bin 31 in HIST overflows. Reset internally in SETUP state.
1 B30_OVF	Bin 30 overflow It is set if bin 30 in HIST overflows Reset internally in SETUP state.
2 B29_OVF	Bin 29 overflow It is set if bin 29 in HIST overflows. Reset internally in SETUP state.
3 B28_OVF	Bin 28 overflow It is set if bin 28 in HIST overflows. Reset internally in SETUP state.
4 B27_OVF	Bin 27 overflow It is set if bin 27 in HIST overflows. Reset internally in SETUP state.
5 B26_OVF	Bin 26 overflow It is set if bin 26 in HIST overflows. Reset internally in SETUP state.
6 B25_OVF	Bin 25 overflow It is set if bin 25 in HIST overflows. Reset internally in SETUP state.
7 B24_OVF	Bin 24 overflow It is set if bin 24 in HIST overflows. Reset internally in SETUP state.
8 B23_OVF	Bin 23 overflow It is set if bin 23 in HIST overflows. Reset internally in SETUP state.
9 B22_OVF	Bin 22 overflow It is set if bin 22 in HIST overflows. Reset internally in SETUP state.

Table continues on the next page...

SPT_HIST_OVF_STATUS0 field descriptions (continued)

Field	Description
10 B21_OVF	Bin 21 overflow It is set if bin 21 in HIST overflows. Reset internally in SETUP state.
11 B20_OVF	Bin 20 overflow It is set if bin 20 in HIST overflows. Reset internally in SETUP state.
12 B19_OVF	Bin 19 overflow It is set if bin 19 in HIST overflows. Reset internally in SETUP state.
13 B18_OVF	Bin 18 overflow It is set if bin 18 in HIST overflows. Reset internally in SETUP state.
14 B17_OVF	Bin 17 overflow It is set if bin 17 in HIST overflows. Reset internally in SETUP state.
15 B16_OVF	Bin 16 overflow It is set if bin 16 in HIST overflows. Reset internally in SETUP state.
16 B15_OVF	Bin 15 overflow It is set if bin 15 in HIST overflows. Reset internally in SETUP state.
17 B14_OVF	Bin 14 overflow It is set if bin 14 in HIST overflows. Reset internally in SETUP state.
18 B13_OVF	Bin 13 overflow It is set if bin 13 in HIST overflows. Reset internally in SETUP state.
19 B12_OVF	Bin 12 overflow It is set if bin 12 in HIST overflows. Reset internally in SETUP state.
20 B11_OVF	Bin 11 overflow It is set if bin 11 in HIST overflows. Reset internally in SETUP state.
21 B10_OVF	Bin 10 overflow It is set if bin 10 in HIST overflows. Reset internally in SETUP state.
22 B9_OVF	Bin 9 overflow It is set if bin 9 in HIST overflows. Reset internally in SETUP state.
23 B8_OVF	Bin 8 overflow It is set if bin 8 in HIST overflows. Reset internally in SETUP state.
24 B7_OVF	Bin 7 overflow It is set if bin 7 in HIST overflows. Reset internally in SETUP state.
25 B6_OVF	Bin 6 overflow It is set if bin 6 in HIST overflows. Reset internally in SETUP state.
26 B5_OVF	Bin 5 overflow

Table continues on the next page...

SPT_HIST_OVF_STATUS0 field descriptions (continued)

Field	Description
	It is set if bin 5 in HIST overflows. Reset internally in SETUP state.
27 B4_OVF	Bin 4 overflow It is set if bin 4 in HIST overflows. Reset internally in SETUP state.
28 B3_OVF	Bin 3 overflow It is set if bin 3 in HIST overflows. Reset internally in SETUP state.
29 B2_OVF	Bin 2 overflow It is set if bin 2 in HIST overflows. Reset internally in SETUP state.
30 B1_OVF	Bin 1 overflow It is set if bin 1 in HIST overflows. Reset internally in SETUP state.
31 B0_OVF	Bin 0 overflow It is set if bin 0 in HIST overflows. Reset internally in SETUP state.

45.8.86 HIST Overflow Status1 Register (SPT_HIST_OVF_STATUS1)

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	B63_OVF	B62_OVF	B61_OVF	B60_OVF	B59_OVF	B58_OVF	B57_OVF	B56_OVF	B55_OVF	B54_OVF	B53_OVF	B52_OVF	B51_OVF	B50_OVF	B49_OVF	B48_OVF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B47_OVF	B46_OVF	B45_OVF	B44_OVF	B43_OVF	B42_OVF	B41_OVF	B40_OVF	B39_OVF	B38_OVF	B37_OVF	B36_OVF	B35_OVF	B34_OVF	B33_OVF	B32_OVF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_HIST_OVF_STATUS1 field descriptions

Field	Description
0 B63_OVF	Bin 63 overflow It is set if bin 63 in HIST overflows. Reset internally in SETUP state.
1 B62_OVF	Bin 62 overflow It is set if bin 62 in HIST overflows. Reset internally in SETUP state.
2 B61_OVF	Bin 61 overflow It is set if bin 61 in HIST overflows. Reset internally in SETUP state.

Table continues on the next page...

SPT_HIST_OVF_STATUS1 field descriptions (continued)

Field	Description
3 B60_OVF	Bin 60 overflow It is set if bin 60 in HIST overflows. Reset internally in SETUP state.
4 B59_OVF	Bin 59 overflow It is set if bin 59 in HIST overflows. Reset internally in SETUP state.
5 B58_OVF	Bin 58 overflow It is set if bin 58 in HIST overflows. Reset internally in SETUP state.
6 B57_OVF	Bin 57 overflow It is set if bin 57 in HIST overflows. Reset internally in SETUP state.
7 B56_OVF	Bin 56 overflow It is set if bin 56 in HIST overflows. Reset internally in SETUP state.
8 B55_OVF	Bin 55 overflow It is set if bin 55 in HIST overflows. Reset internally in SETUP state.
9 B54_OVF	Bin 54 overflow It is set if bin 54 in HIST overflows. Reset internally in SETUP state.
10 B53_OVF	Bin 53 overflow It is set if bin 53 in HIST overflows. Reset internally in SETUP state.
11 B52_OVF	Bin 52 overflow It is set if bin 52 in HIST overflows. Reset internally in SETUP state.
12 B51_OVF	Bin 51 overflow It is set if bin 51 in HIST overflows. Reset internally in SETUP state.
13 B50_OVF	Bin 50 overflow It is set if bin 50 in HIST overflows. Reset internally in SETUP state.
14 B49_OVF	Bin 49 overflow It is set if bin 49 in HIST overflows. Reset internally in SETUP state.
15 B48_OVF	Bin 48 overflow It is set if bin 48 in HIST overflows. Reset internally in SETUP state.
16 B47_OVF	Bin 47 overflow It is set if bin 47 in HIST overflows. Reset internally in SETUP state.
17 B46_OVF	Bin 46 overflow It is set if bin 46 in HIST overflows. Reset internally in SETUP state.
18 B45_OVF	Bin 45 overflow It is set if bin 45 in HIST overflows. Reset internally in SETUP state.
19 B44_OVF	Bin 44 overflow

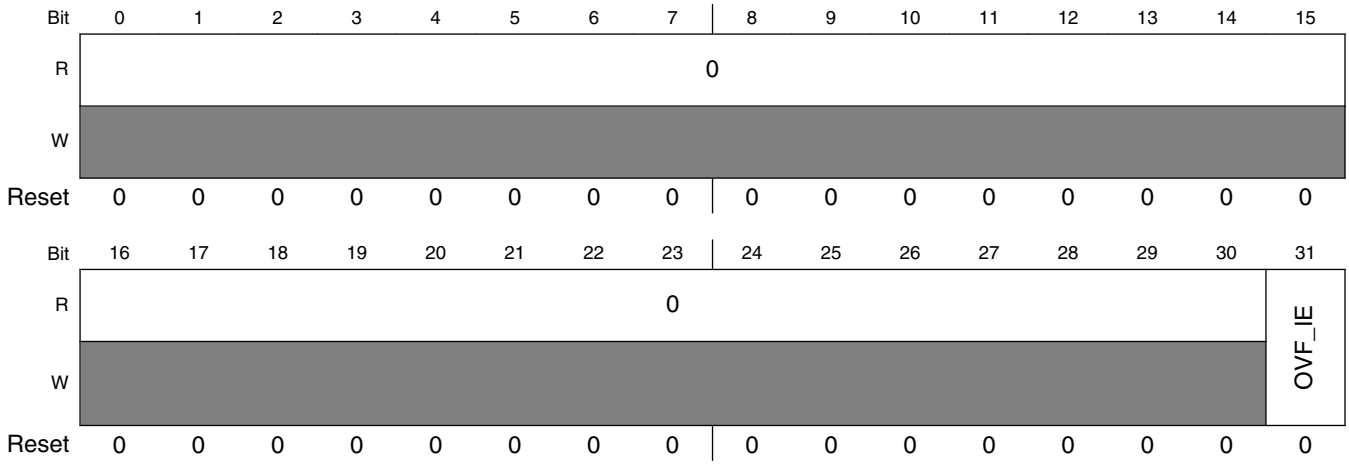
Table continues on the next page...

SPT_HIST_OVF_STATUS1 field descriptions (continued)

Field	Description
	It is set if bin 44 in HIST overflows. Reset internally in SETUP state.
20 B43_OVF	Bin 43 overflow It is set if bin 43 in HIST overflows. Reset internally in SETUP state.
21 B42_OVF	Bin 42 overflow It is set if bin 42 in HIST overflows. Reset internally in SETUP state.
22 B41_OVF	Bin 41 overflow It is set if bin 41 in HIST overflows. Reset internally in SETUP state.
23 B40_OVF	Bin 40 overflow It is set if bin 40 in HIST overflows. Reset internally in SETUP state.
24 B39_OVF	Bin 39 overflow It is set if bin 39 in HIST overflows. Reset internally in SETUP state.
25 B38_OVF	Bin 38 overflow It is set if bin 38 in HIST overflows. Reset internally in SETUP state.
26 B37_OVF	Bin 37 overflow It is set if bin 37 in HIST overflows. Reset internally in SETUP state.
27 B36_OVF	Bin 36 overflow It is set if bin 36 in HIST overflows. Reset internally in SETUP state.
28 B35_OVF	Bin 35 overflow It is set if bin 35 in HIST overflows. Reset internally in SETUP state.
29 B34_OVF	Bin 34 overflow It is set if bin 34 in HIST overflows. Reset internally in SETUP state.
30 B33_OVF	Bin 33 overflow It is set if bin 33 in HIST overflows. Reset internally in SETUP state.
31 B32_OVF	Bin 32 overflow It is set if bin 32 in HIST overflows. Reset internally in SETUP state.

45.8.87 HIST Overflow Interrupt Enable Register (SPT_HIST_OVF_IE)

Address: 0h base + 228h offset = 228h

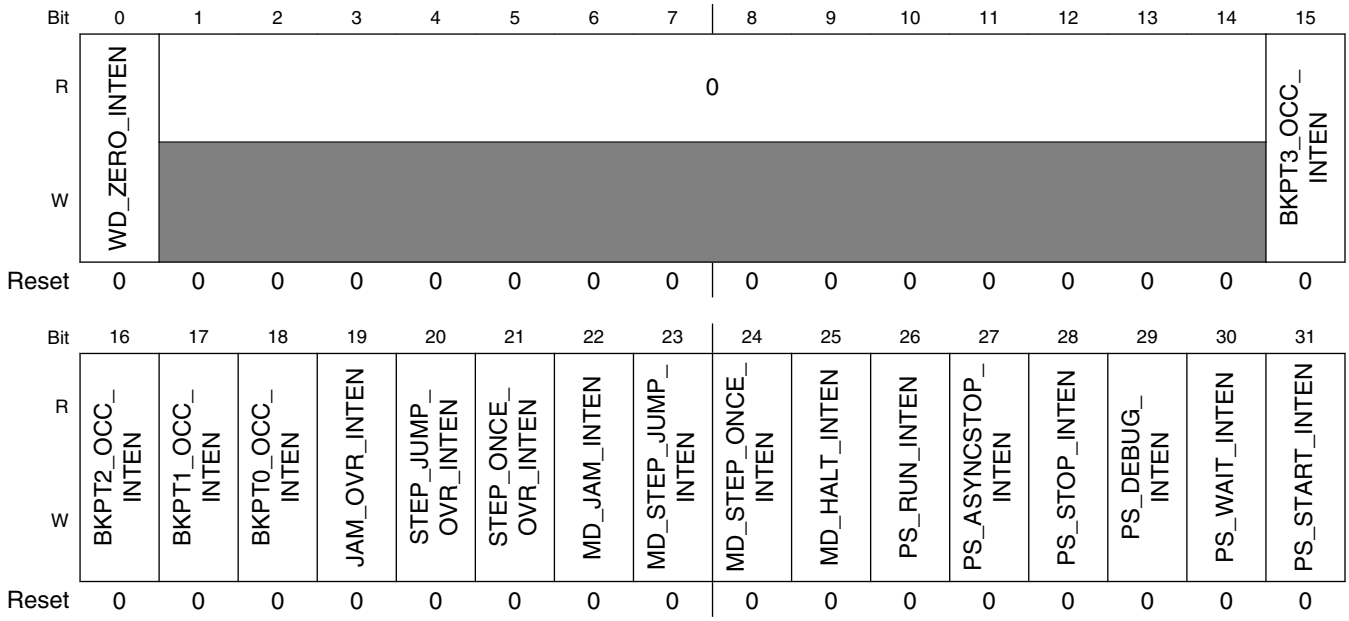


SPT_HIST_OVF_IE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 OVF_IE	HIST overflow interrupt enable If set then an interrupt is caused when any bit of SPT_HIST_OVF_STATUS0 and SPT_HIST_OVF_STATUS1 registers is set.

45.8.88 Interrupt Enable Register 0 (SPT_CS_INTEN0)

Address: 0h base + 22Ch offset = 22Ch



SPT_CS_INTEN0 field descriptions

Field	Description
0 WD_ZERO_INTEN	Watchdog reached zero interrupt enable If set then an interrupt is issued when the watchdog reaches zero in the count-down mode
1–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BKPT3_OCC_INTEN	BKPT3 interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[BKPT3_OCC] is set
16 BKPT2_OCC_INTEN	BKPT2 interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[BKPT2_OCC] is set
17 BKPT1_OCC_INTEN	BKPT1 interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[BKPT1_OCC] is set
18 BKPT0_OCC_INTEN	BKPT0 interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[BKPT0_OCC] is set
19 JAM_OVR_INTEN	Jamming instruction over interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[JAM_OVR] gets set upon instruction completion
20 STEP_JUMP_OVR_INTEN	'Breakpoint hit in STEP JUMP mode' interrupt enable

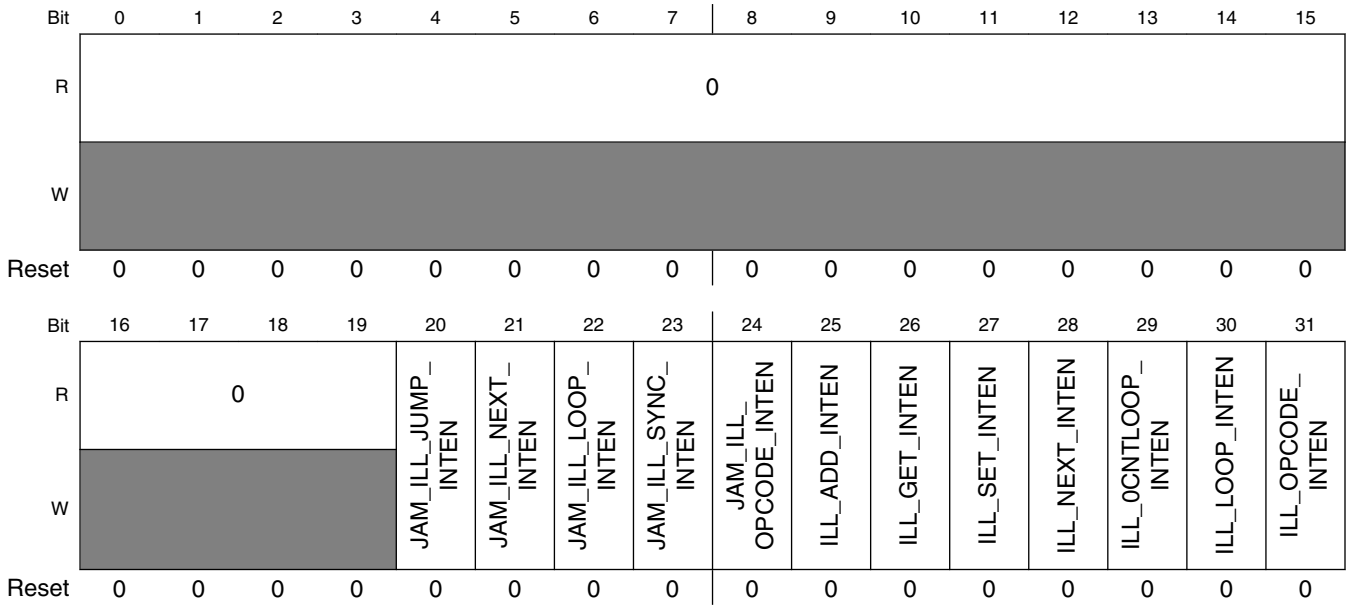
Table continues on the next page...

SPT_CS_INTEN0 field descriptions (continued)

Field	Description
	If set then an interrupt is issued when SPT_CS_STATUS0[STEP_JUMP_OVR] gets set on hitting a breakpoint
21 STEP_ONCE_ OVR_INTEN	STEP ONCE mode instruction over interrupt enable If set then an interrupt is issued when SPT_CS_STATUS0[STEP_ONCE_OVR] gets set upon instruction completion
22 MD_JAM_INTEN	Interrupt enable for instruction jamming mode If set then an interrupt is issued when SPT_CS_STATUS0[MD_JAM] gets set.
23 MD_STEP_ JUMP_INTEN	Interrupt enable for STEP JUMP mode If set then an interrupt is issued when SPT_CS_STATUS0[MD_STEP_JUMP] gets set.
24 MD_STEP_ ONCE_INTEN	Interrupt enable for STEP ONCE mode If set then an interrupt is issued when SPT_CS_STATUS0[MD_STEP_ONCE] gets set.
25 MD_HALT_ INTEN	Interrupt enable for HALT mode If set then an interrupt is issued when SPT_CS_STATUS0[MD_HALT] gets set.
26 PS_RUN_INTEN	Interrupt enable for RUN state If set then an interrupt is issued when SPT_CS_STATUS0[PS_RUN] gets set.
27 PS_ ASYNCSTOP_ INTEN	Interrupt enable for ASYNCSTOP state If set then an interrupt is issued when SPT_CS_STATUS0[PS_ASYNCSTOP] gets set.
28 PS_STOP_ INTEN	Interrupt enable for STOP state If set then an interrupt is issued when SPT_CS_STATUS0[PS_STOP] gets set.
29 PS_DEBUG_ INTEN	Interrupt enable for DEBUG state If set then an interrupt is issued when SPT_CS_STATUS0[PS_DEBUG] gets set.
30 PS_WAIT_INTEN	Interrupt enable for WAIT state If set then an interrupt is issued when SPT_CS_STATUS0[PS_WAIT] gets set.
31 PS_START_ INTEN	Interrupt enable for START state If set then an interrupt is issued when SPT_CS_STATUS0[PS_START] gets set.

45.8.89 Interrupt Enable Register 1 (SPT_CS_INTEN1)

Address: 0h base + 230h offset = 230h



SPT_CS_INTEN1 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 JAM_ILL_JUMP_INTEN	Illegal JUMP in jamming mode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[JAM_ILL_JUMP] bit is set.
21 JAM_ILL_NEXT_INTEN	Illegal NEXT in jamming mode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[JAM_ILL_NEXT] bit is set.
22 JAM_ILL_LOOP_INTEN	Illegal LOOP in jamming mode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[JAM_ILL_LOOP] bit is set.
23 JAM_ILL_SYNC_INTEN	Illegal SYNC in jamming mode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[JAM_ILL_SYNC] bit is set.
24 JAM_ILL_OPCODE_INTEN	Illegal Opcode in jamming mode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[JAM_ILL_OPCODE] bit is set.
25 ILL_ADD_INTEN	Illegal ADD interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_ADD] bit is set.
26 ILL_GET_INTEN	Illegal GET interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_GET] bit is set.

Table continues on the next page...

SPT_CS_INTEN1 field descriptions (continued)

Field	Description
27 ILL_SET_INTEN	Illegal SET interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_SET] bit is set.
28 ILL_NEXT_ INTEN	Illegal NEXT interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_NEXT] bit is set.
29 ILL_0CNTLOOP_ INTEN	Illegal loop count error interrupt If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_0CNTLOOP] bit is set.
30 ILL_LOOP_ INTEN	Illegal LOOP interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_LOOP] bit is set.
31 ILL_OPCODE_ INTEN	Illegal Opcode interrupt enable If set then an interrupt is raised when the SPT_CS_STATUS1[ILL_OPCODE] bit is set.

45.8.90 EVT1 Interrupt Enable Register (SPT_CS_EVT1_INTEN)

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																EVT1_INTEN															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_EVT1_INTEN field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 EVT1_INTEN	EVT interrupt enable for CPU core 1 If a bit is set then an interrupt is issued if the corresponding bit in SPT_CS_EVTREG1[EVTREG1] gets set

45.8.91 EVT2 Interrupt Enable Register (SPT_CS_EVT2_INTEN)

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																EVT2_INTEN															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_CS_EVT2_INTEN field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 EVT2_INTEN	EVT interrupt enable for CPU core 2 If a bit is set then an interrupt is issued if the corresponding bit in SPT_CS_EVTREG2[EVTREG2] gets set

45.8.92 SPT_WR_0_15_CTRL_REG

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WR15_ACC_CTRL	WR15_LCK	WR14_ACC_CTRL	WR14_LCK	WR13_ACC_CTRL	WR13_LCK	WR12_ACC_CTRL	WR12_LCK	WR11_ACC_CTRL	WR11_LCK	WR10_ACC_CTRL	WR10_LCK	WR9_ACC_CTRL	WR9_LCK	WR8_ACC_CTRL	WR8_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WR7_ACC_CTRL	WR7_LCK	WR6_ACC_CTRL	WR6_LCK	WR5_ACC_CTRL	WR5_LCK	WR4_ACC_CTRL	WR4_LCK	WR3_ACC_CTRL	WR3_LCK	WR2_ACC_CTRL	WR2_LCK	WR1_ACC_CTRL	WR1_LCK	WR0_ACC_CTRL	WR0_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_WR_0_15_CTRL_REG field descriptions

Field	Description
0 WR15_ACC_CTRL	Work register 15 access control 0: SPT has write access to WR_R15_RE and WR_R15_IM 1: CPU has write access to WR_R15_RE and WR_R15_IM
1 WR15_LCK	Work register 15 lock 1: Writes to WR_R15_RE and WR_R15_IM are locked. These registers written by either CPU or SPT as defined by WR15_ACC_CTRL bit. 0: Writes to WR_R15_RE and WR_R15_IM are unlocked. These registers can be written by CPU and SPT
2 WR14_ACC_CTRL	Work register 14 access control 0: SPT has write access to WR_R14_RE and WR_R14_IM 1: CPU has write access to WR_R14_RE and WR_R14_IM
3 WR14_LCK	Work register 14 lock

Table continues on the next page...

SPT_WR_0_15_CTRL_REG field descriptions (continued)

Field	Description
	1: Writes to WR_R14_RE and WR_R14_IM are locked. These registers written by either CPU or SPT as defined by WR14_ACC_CTRL bit. 0: Writes to WR_R14_RE and WR_R14_IM are unlocked. These registers can be written by CPU and SPT
4 WR13_ACC_CTRL	Work register 13 access control 0: SPT has write access to WR_R13_RE and WR_R13_IM 1: CPU has write access to WR_R13_RE and WR_R13_IM
5 WR13_LCK	Work register 13 lock 1: Writes to WR_R13_RE and WR_R13_IM are locked. These registers written by either CPU or SPT as defined by WR13_ACC_CTRL bit. 0: Writes to WR_R13_RE and WR_R13_IM are unlocked. These registers can be written by CPU and SPT
6 WR12_ACC_CTRL	Work register 12 access control 0: SPT has write access to WR_R12_RE and WR_R12_IM 1: CPU has write access to WR_R12_RE and WR_R12_IM
7 WR12_LCK	Work register 12 lock 1: Writes to WR_R12_RE and WR_R12_IM are locked. These registers written by either CPU or SPT as defined by WR12_ACC_CTRL bit. 0: Writes to WR_R12_RE and WR_R12_IM are unlocked. These registers can be written by CPU and SPT
8 WR11_ACC_CTRL	Work register 11 access control 0: SPT has write access to WR_R11_RE and WR_R11_IM 1: CPU has write access to WR_R11_RE and WR_R11_IM
9 WR11_LCK	Work register 11 lock 1: Writes to WR_R11_RE and WR_R11_IM are locked. These registers written by either CPU or SPT as defined by WR11_ACC_CTRL bit. 0: Writes to WR_R11_RE and WR_R11_IM are unlocked. These registers can be written by CPU and SPT
10 WR10_ACC_CTRL	Work register 10 access control 0: SPT has write access to WR_R10_RE and WR_R10_IM 1: CPU has write access to WR_R10_RE and WR_R10_IM
11 WR10_LCK	Work register 10 lock 1: Writes to WR_R10_RE and WR_R10_IM are locked. These registers written by either CPU or SPT as defined by WR10_ACC_CTRL bit. 0: Writes to WR_R10_RE and WR_R10_IM are unlocked. These registers can be written by CPU and SPT
12 WR9_ACC_CTRL	Work register 9 access control 0: SPT has write access to WR_R9_RE and WR_R9_IM 1: CPU has write access to WR_R9_RE and WR_R9_IM

Table continues on the next page...

SPT_WR_0_15_CTRL_REG field descriptions (continued)

Field	Description
13 WR9_LCK	Work register 9 lock 1: Writes to WR_R9_RE and WR_R9_IM are locked. These registers written by either CPU or SPT as defined by WR9_ACC_CTRL bit. 0: Writes to WR_R9_RE and WR_R9_IM are unlocked. These registers can be written by CPU and SPT
14 WR8_ACC_ CTRL	Work register 8 access control 0: SPT has write access to WR_R8_RE and WR_R8_IM 1: CPU has write access to WR_R8_RE and WR_R8_IM
15 WR8_LCK	Work register 8 lock 1: Writes to WR_R8_RE and WR_R8_IM are locked. These registers written by either CPU or SPT as defined by WR8_ACC_CTRL bit. 0: Writes to WR_R8_RE and WR_R8_IM are unlocked. These registers can be written by CPU and SPT
16 WR7_ACC_ CTRL	Work register 7 access control 0: SPT has write access to WR_R7_RE and WR_R7_IM 1: CPU has write access to WR_R7_RE and WR_R7_IM
17 WR7_LCK	Work register 7 lock 1: Writes to WR_R7_RE and WR_R7_IM are locked. These registers written by either CPU or SPT as defined by WR7_ACC_CTRL bit. 0: Writes to WR_R7_RE and WR_R7_IM are unlocked. These registers can be written by CPU and SPT
18 WR6_ACC_ CTRL	Work register 6 access control 0: SPT has write access to WR_R6_RE and WR_R6_IM 1: CPU has write access to WR_R6_RE and WR_R6_IM
19 WR6_LCK	Work register 6 lock 1: Writes to WR_R6_RE and WR_R6_IM are locked. These registers written by either CPU or SPT as defined by WR6_ACC_CTRL bit. 0: Writes to WR_R6_RE and WR_R6_IM are unlocked. These registers can be written by CPU and SPT
20 WR5_ACC_ CTRL	Work register 5 access control 0: SPT has write access to WR_R5_RE and WR_R5_IM 1: CPU has write access to WR_R5_RE and WR_R5_IM
21 WR5_LCK	Work register 5 lock 1: Writes to WR_R5_RE and WR_R5_IM are locked. These registers written by either CPU or SPT as defined by WR5_ACC_CTRL bit. 0: Writes to WR_R5_RE and WR_R5_IM are unlocked. These registers can be written by CPU and SPT
22 WR4_ACC_ CTRL	Work register 4 access control 0: SPT has write access to WR_R4_RE and WR_R4_IM 1: CPU has write access to WR_R4_RE and WR_R4_IM
23 WR4_LCK	Work register 4 lock 1: Writes to WR_R4_RE and WR_R4_IM are locked. These registers written by either CPU or SPT as defined by WR4_ACC_CTRL bit.

Table continues on the next page...

SPT_WR_0_15_CTRL_REG field descriptions (continued)

Field	Description
	0: Writes to WR_R4_RE and WR_R4_IM are unlocked. These registers can be written by CPU and SPT
24 WR3_ACC_ CTRL	Work register 3 access control 0: SPT has write access to WR_R3_RE and WR_R3_IM 1: CPU has write access to WR_R3_RE and WR_R3_IM
25 WR3_LCK	Work register 3 lock 1: Writes to WR_R3_RE and WR_R3_IM are locked. These registers written by either CPU or SPT as defined by WR3_ACC_CTRL bit. 0: Writes to WR_R3_RE and WR_R3_IM are unlocked. These registers can be written by CPU and SPT
26 WR2_ACC_ CTRL	Work register 2 access control 0: SPT has write access to WR_R2_RE and WR_R2_IM 1: CPU has write access to WR_R2_RE and WR_R2_IM
27 WR2_LCK	Work register 2 lock 1: Writes to WR_R2_RE and WR_R2_IM are locked. These registers written by either CPU or SPT as defined by WR2_ACC_CTRL bit. 0: Writes to WR_R2_RE and WR_R2_IM are unlocked. These registers can be written by CPU and SPT
28 WR1_ACC_ CTRL	Work register 1 access control 0: SPT has write access to WR_R1_RE and WR_R1_IM 1: CPU has write access to WR_R1_RE and WR_R1_IM
29 WR1_LCK	Work register 1 lock 1: Writes to WR_R1_RE and WR_R1_IM are locked. These registers written by either CPU or SPT as defined by WR1_ACC_CTRL bit. 0: Writes to WR_R1_RE and WR_R1_IM are unlocked. These registers can be written by CPU and SPT
30 WR0_ACC_ CTRL	Work register 0 access control 0: SPT has write access to WR_R0_RE and WR_R0_IM 1: CPU has write access to WR_R0_RE and WR_R0_IM
31 WR0_LCK	Work register 0 lock 1: Writes to WR_R0_RE and WR_R0_IM are locked. These registers written by either CPU or SPT as defined by WR0_ACC_CTRL bit. 0: Writes to WR_R0_RE and WR_R0_IM are unlocked. These registers can be written by CPU and SPT

45.8.93 SPT_WR_16_31_CTRL_REG

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WR31_ACC_CTRL	WR31_LCK	WR30_ACC_CTRL	WR30_LCK	WR29_ACC_CTRL	WR29_LCK	WR28_ACC_CTRL	WR28_LCK	WR27_ACC_CTRL	WR27_LCK	WR26_ACC_CTRL	WR26_LCK	WR25_ACC_CTRL	WR25_LCK	WR24_ACC_CTRL	WR24_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WR23_ACC_CTRL	WR23_LCK	WR22_ACC_CTRL	WR22_LCK	WR21_ACC_CTRL	WR21_LCK	WR20_ACC_CTRL	WR20_LCK	WR19_ACC_CTRL	WR19_LCK	WR18_ACC_CTRL	WR18_LCK	WR17_ACC_CTRL	WR17_LCK	WR16_ACC_CTRL	WR16_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_WR_16_31_CTRL_REG field descriptions

Field	Description
0 WR31_ACC_CTRL	Work register 31 access control 0: SPT has write access to WR_R31_RE and WR_R31_IM 1: CPU has write access to WR_R31_RE and WR_R31_IM
1 WR31_LCK	Work register 31 lock 1: Writes to WR_R31_RE and WR_R31_IM are locked. These registers written by either CPU or SPT as defined by WR31_ACC_CTRL bit. 0: Writes to WR_R31_RE and WR_R31_IM are unlocked. These registers can be written by CPU and SPT
2 WR30_ACC_CTRL	Work register 30 access control 0: SPT has write access to WR_R30_RE and WR_R30_IM 1: CPU has write access to WR_R30_RE and WR_R30_IM
3 WR30_LCK	Work register 30 lock 1: Writes to WR_R30_RE and WR_R30_IM are locked. These registers written by either CPU or SPT as defined by WR30_ACC_CTRL bit. 0: Writes to WR_R30_RE and WR_R30_IM are unlocked. These registers can be written by CPU and SPT
4 WR29_ACC_CTRL	Work register 29 access control 0: SPT has write access to WR_R29_RE and WR_R29_IM 1: CPU has write access to WR_R29_RE and WR_R29_IM

Table continues on the next page...

SPT_WR_16_31_CTRL_REG field descriptions (continued)

Field	Description
5 WR29_LCK	Work register 29 lock 1: Writes to WR_R29_RE and WR_R29_IM are locked. These registers written by either CPU or SPT as defined by WR29_ACC_CTRL bit. 0: Writes to WR_R29_RE and WR_R29_IM are unlocked. These registers can be written by CPU and SPT
6 WR28_ACC_ CTRL	Work register 28 access control 0: SPT has write access to WR_R28_RE and WR_R28_IM 1: CPU has write access to WR_R28_RE and WR_R28_IM
7 WR28_LCK	Work register 28 lock 1: Writes to WR_R28_RE and WR_R28_IM are locked. These registers written by either CPU or SPT as defined by WR28_ACC_CTRL bit. 0: Writes to WR_R28_RE and WR_R28_IM are unlocked. These registers can be written by CPU and SPT
8 WR27_ACC_ CTRL	Work register 27 access control 0: SPT has write access to WR_R27_RE and WR_R27_IM 1: CPU has write access to WR_R27_RE and WR_R27_IM
9 WR27_LCK	Work register 27 lock 1: Writes to WR_R27_RE and WR_R27_IM are locked. These registers written by either CPU or SPT as defined by WR27_ACC_CTRL bit. 0: Writes to WR_R27_RE and WR_R27_IM are unlocked. These registers can be written by CPU and SPT
10 WR26_ACC_ CTRL	Work register 26 access control 0: SPT has write access to WR_R26_RE and WR_R26_IM 1: CPU has write access to WR_R26_RE and WR_R26_IM
11 WR26_LCK	Work register 26 lock 1: Writes to WR_R26_RE and WR_R26_IM are locked. These registers written by either CPU or SPT as defined by WR26_ACC_CTRL bit. 0: Writes to WR_R26_RE and WR_R26_IM are unlocked. These registers can be written by CPU and SPT
12 WR25_ACC_ CTRL	Work register 25 access control 0: SPT has write access to WR_R25_RE and WR_R25_IM 1: CPU has write access to WR_R25_RE and WR_R25_IM
13 WR25_LCK	Work register 25 lock 1: Writes to WR_R25_RE and WR_R25_IM are locked. These registers written by either CPU or SPT as defined by WR25_ACC_CTRL bit. 0: Writes to WR_R25_RE and WR_R25_IM are unlocked. These registers can be written by CPU and SPT
14 WR24_ACC_ CTRL	Work register 24 access control 0: SPT has write access to WR_R24_RE and WR_R24_IM

Table continues on the next page...

SPT_WR_16_31_CTRL_REG field descriptions (continued)

Field	Description
	1: CPU has write access to WR_R24_RE and WR_R24_IM
15 WR24_LCK	Work register 24 lock 1: Writes to WR_R24_RE and WR_R24_IM are locked. These registers written by either CPU or SPT as defined by WR24_ACC_CTRL bit. 0: Writes to WR_R24_RE and WR_R24_IM are unlocked. These registers can be written by CPU and SPT
16 WR23_ACC_ CTRL	Work register 23 access control 0: SPT has write access to WR_R23_RE and WR_R23_IM 1: CPU has write access to WR_R23_RE and WR_R23_IM
17 WR23_LCK	Work register 23 lock 1: Writes to WR_R23_RE and WR_R23_IM are locked. These registers written by either CPU or SPT as defined by WR23_ACC_CTRL bit. 0: Writes to WR_R23_RE and WR_R23_IM are unlocked. These registers can be written by CPU and SPT
18 WR22_ACC_ CTRL	Work register 22 access control 0: SPT has write access to WR_R22_RE and WR_R22_IM 1: CPU has write access to WR_R22_RE and WR_R22_IM
19 WR22_LCK	Work register 22 lock 1: Writes to WR_R22_RE and WR_R22_IM are locked. These registers written by either CPU or SPT as defined by WR22_ACC_CTRL bit. 0: Writes to WR_R22_RE and WR_R22_IM are unlocked. These registers can be written by CPU and SPT
20 WR21_ACC_ CTRL	Work register 21 access control 0: SPT has write access to WR_R21_RE and WR_R21_IM 1: CPU has write access to WR_R21_RE and WR_R21_IM
21 WR21_LCK	Work register 21 lock 1: Writes to WR_R21_RE and WR_R21_IM are locked. These registers written by either CPU or SPT as defined by WR21_ACC_CTRL bit. 0: Writes to WR_R21_RE and WR_R21_IM are unlocked. These registers can be written by CPU and SPT
22 WR20_ACC_ CTRL	Work register 20 access control 0: SPT has write access to WR_R20_RE and WR_R20_IM 1: CPU has write access to WR_R20_RE and WR_R20_IM
23 WR20_LCK	Work register 20 lock 1: Writes to WR_R20_RE and WR_R20_IM are locked. These registers written by either CPU or SPT as defined by WR20_ACC_CTRL bit. 0: Writes to WR_R20_RE and WR_R20_IM are unlocked. These registers can be written by CPU and SPT

Table continues on the next page...

SPT_WR_16_31_CTRL_REG field descriptions (continued)

Field	Description
24 WR19_ACC_ CTRL	Work register 19 access control 0: SPT has write access to WR_R19_RE and WR_R19_IM 1: CPU has write access to WR_R19_RE and WR_R19_IM
25 WR19_LCK	Work register 19 lock 1: Writes to WR_R19_RE and WR_R19_IM are locked. These registers written by either CPU or SPT as defined by WR19_ACC_CTRL bit. 0: Writes to WR_R19_RE and WR_R19_IM are unlocked. These registers can be written by CPU and SPT
26 WR18_ACC_ CTRL	Work register 18 access control 0: SPT has write access to WR_R18_RE and WR_R18_IM 1: CPU has write access to WR_R18_RE and WR_R18_IM
27 WR18_LCK	Work register 18 lock 1: Writes to WR_R18_RE and WR_R18_IM are locked. These registers written by either CPU or SPT as defined by WR18_ACC_CTRL bit. 0: Writes to WR_R18_RE and WR_R18_IM are unlocked. These registers can be written by CPU and SPT
28 WR17_ACC_ CTRL	Work register 1 access control 0: SPT has write access to WR_R17_RE and WR_R17_IM 1: CPU has write access to WR_R17_RE and WR_R17_IM
29 WR17_LCK	Work register 17 lock 1: Writes to WR_R17_RE and WR_R1_IM are locked. These registers written by either CPU or SPT as defined by WR17_ACC_CTRL bit. 0: Writes to WR_R17_RE and WR_R1_IM are unlocked. These registers can be written by CPU and SPT
30 WR16_ACC_ CTRL	Work register 15 access control 0: SPT has write access to WR_R16_RE and WR_R16_IM 1: CPU has write access to WR_R16_RE and WR_R16_IM
31 WR16_LCK	Work register 15 lock 1: Writes to WR_R16_RE and WR_R16_IM are locked. These registers written by either CPU or SPT as defined by WR16_ACC_CTRL bit. 0: Writes to WR_R16_RE and WR_R16_IM are unlocked. These registers can be written by CPU and SPT

45.8.94 SPT_WR_32_47_CTRL_REG

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WR47_ACC_CTRL	WR47_LCK	WR46_ACC_CTRL	WR46_LCK	WR45_ACC_CTRL	WR45_LCK	WR44_ACC_CTRL	WR44_LCK	WR43_ACC_CTRL	WR43_LCK	WR42_ACC_CTRL	WR42_LCK	WR41_ACC_CTRL	WR41_LCK	WR40_ACC_CTRL	WR40_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WR39_ACC_CTRL	WR39_LCK	WR38_ACC_CTRL	WR38_LCK	WR37_ACC_CTRL	WR37_LCK	WR36_ACC_CTRL	WR36_LCK	WR35_ACC_CTRL	WR35_LCK	WR34_ACC_CTRL	WR34_LCK	WR33_ACC_CTRL	WR33_LCK	WR32_ACC_CTRL	WR32_LCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPT_WR_32_47_CTRL_REG field descriptions

Field	Description
0 WR47_ACC_CTRL	Work register 47 access control 0: SPT has write access to WR_R47_RE and WR_R47_IM 1: CPU has write access to WR_R47_RE and WR_R47_IM
1 WR47_LCK	Work register 31 lock 1: Writes to WR_R47_RE and WR_R47_IM are locked. These registers written by either CPU or SPT as defined by WR47_ACC_CTRL bit. 0: Writes to WR_R47_RE and WR_R47_IM are unlocked. These registers can be written by CPU and SPT
2 WR46_ACC_CTRL	Work register 46 access control 0: SPT has write access to WR_R46_RE and WR_R46_IM 1: CPU has write access to WR_R46_RE and WR_R46_IM
3 WR46_LCK	Work register 46 lock 1: Writes to WR_R46_RE and WR_R46_IM are locked. These registers written by either CPU or SPT as defined by WR46_ACC_CTRL bit. 0: Writes to WR_R46_RE and WR_R46_IM are unlocked. These registers can be written by CPU and SPT
4 WR45_ACC_CTRL	Work register 45 access control 0: SPT has write access to WR_R45_RE and WR_R45_IM 1: CPU has write access to WR_R45_RE and WR_R45_IM

Table continues on the next page...

SPT_WR_32_47_CTRL_REG field descriptions (continued)

Field	Description
5 WR45_LCK	Work register 45 lock 1: Writes to WR_R45_RE and WR_R45_IM are locked. These registers written by either CPU or SPT as defined by WR45_ACC_CTRL bit. 0: Writes to WR_R45_RE and WR_R45_IM are unlocked. These registers can be written by CPU and SPT
6 WR44_ACC_ CTRL	Work register 44 access control 0: SPT has write access to WR_R44_RE and WR_R44_IM 1: CPU has write access to WR_R44_RE and WR_R44_IM
7 WR44_LCK	Work register 44 lock 1: Writes to WR_R44_RE and WR_R44_IM are locked. These registers written by either CPU or SPT as defined by WR44_ACC_CTRL bit. 0: Writes to WR_R44_RE and WR_R44_IM are unlocked. These registers can be written by CPU and SPT
8 WR43_ACC_ CTRL	Work register 43 access control 0: SPT has write access to WR_R43_RE and WR_R43_IM 1: CPU has write access to WR_R43_RE and WR_R43_IM
9 WR43_LCK	Work register 43 lock 1: Writes to WR_R43_RE and WR_R43_IM are locked. These registers written by either CPU or SPT as defined by WR43_ACC_CTRL bit. 0: Writes to WR_R43_RE and WR_R43_IM are unlocked. These registers can be written by CPU and SPT
10 WR42_ACC_ CTRL	Work register 42 access control 0: SPT has write access to WR_R42_RE and WR_R42_IM 1: CPU has write access to WR_R42_RE and WR_R42_IM
11 WR42_LCK	Work register 42 lock 1: Writes to WR_R42_RE and WR_R42_IM are locked. These registers written by either CPU or SPT as defined by WR42_ACC_CTRL bit. 0: Writes to WR_R42_RE and WR_R42_IM are unlocked. These registers can be written by CPU and SPT
12 WR41_ACC_ CTRL	Work register 41 access control 0: SPT has write access to WR_R41_RE and WR_R41_IM 1: CPU has write access to WR_R41_RE and WR_R41_IM
13 WR41_LCK	Work register 41 lock 1: Writes to WR_R41_RE and WR_R41_IM are locked. These registers written by either CPU or SPT as defined by WR41_ACC_CTRL bit. 0: Writes to WR_R41_RE and WR_R41_IM are unlocked. These registers can be written by CPU and SPT
14 WR40_ACC_ CTRL	Work register 40 access control 0: SPT has write access to WR_R40_RE and WR_R40_IM

Table continues on the next page...

SPT_WR_32_47_CTRL_REG field descriptions (continued)

Field	Description
	1: CPU has write access to WR_R40_RE and WR_R40_IM
15 WR40_LCK	Work register 40 lock 1: Writes to WR_R40_RE and WR_R40_IM are locked. These registers written by either CPU or SPT as defined by WR40_ACC_CTRL bit. 0: Writes to WR_R40_RE and WR_R40_IM are unlocked. These registers can be written by CPU and SPT
16 WR39_ACC_ CTRL	Work register 39 access control 0: SPT has write access to WR_R39_RE and WR_R39_IM 1: CPU has write access to WR_R39_RE and WR_R39_IM
17 WR39_LCK	Work register 39 lock 1: Writes to WR_R39_RE and WR_R39_IM are locked. These registers written by either CPU or SPT as defined by WR39_ACC_CTRL bit. 0: Writes to WR_R39_RE and WR_R39_IM are unlocked. These registers can be written by CPU and SPT
18 WR38_ACC_ CTRL	Work register 38 access control 0: SPT has write access to WR_R38_RE and WR_R38_IM 1: CPU has write access to WR_R38_RE and WR_R38_IM
19 WR38_LCK	Work register 38 lock 1: Writes to WR_R38_RE and WR_R38_IM are locked. These registers written by either CPU or SPT as defined by WR38_ACC_CTRL bit. 0: Writes to WR_R38_RE and WR_R38_IM are unlocked. These registers can be written by CPU and SPT
20 WR37_ACC_ CTRL	Work register 37 access control 0: SPT has write access to WR_R37_RE and WR_R37_IM 1: CPU has write access to WR_R37_RE and WR_R37_IM
21 WR37_LCK	Work register 37 lock 1: Writes to WR_R37_RE and WR_R37_IM are locked. These registers written by either CPU or SPT as defined by WR37_ACC_CTRL bit. 0: Writes to WR_R37_RE and WR_R37_IM are unlocked. These registers can be written by CPU and SPT
22 WR36_ACC_ CTRL	Work register 36 access control 0: SPT has write access to WR_R36_RE and WR_R36_IM 1: CPU has write access to WR_R36_RE and WR_R36_IM
23 WR36_LCK	Work register 36 lock 1: Writes to WR_R36_RE and WR_R36_IM are locked. These registers written by either CPU or SPT as defined by WR36_ACC_CTRL bit. 0: Writes to WR_R36_RE and WR_R36_IM are unlocked. These registers can be written by CPU and SPT

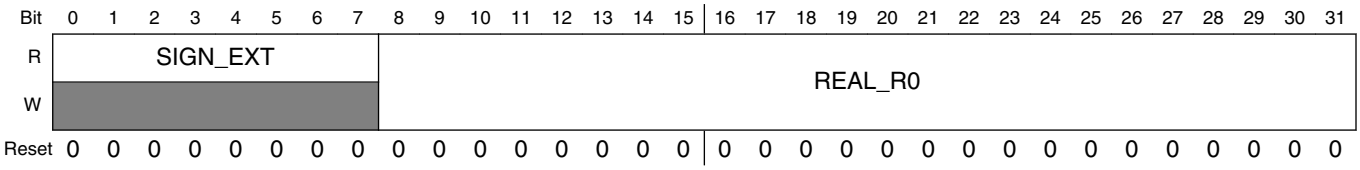
Table continues on the next page...

SPT_WR_32_47_CTRL_REG field descriptions (continued)

Field	Description
24 WR35_ACC_ CTRL	Work register 35 access control 0: SPT has write access to WR_R35_RE and WR_R35_IM 1: CPU has write access to WR_R35_RE and WR_R35_IM
25 WR35_LCK	Work register 35 lock 1: Writes to WR_R35_RE and WR_R35_IM are locked. These registers written by either CPU or SPT as defined by WR35_ACC_CTRL bit. 0: Writes to WR_R35_RE and WR_R35_IM are unlocked. These registers can be written by CPU and SPT
26 WR34_ACC_ CTRL	Work register 34 access control 0: SPT has write access to WR_R34_RE and WR_R34_IM 1: CPU has write access to WR_R34_RE and WR_R34_IM
27 WR34_LCK	Work register 34 lock 1: Writes to WR_R34_RE and WR_R34_IM are locked. These registers written by either CPU or SPT as defined by WR34_ACC_CTRL bit. 0: Writes to WR_R34_RE and WR_R34_IM are unlocked. These registers can be written by CPU and SPT
28 WR33_ACC_ CTRL	Work register 33 access control 0: SPT has write access to WR_R33_RE and WR_R33_IM 1: CPU has write access to WR_R33_RE and WR_R33_IM
29 WR33_LCK	Work register 33 lock 1: Writes to WR_R33_RE and WR_R33_IM are locked. These registers written by either CPU or SPT as defined by WR33_ACC_CTRL bit. 0: Writes to WR_R33_RE and WR_R33_IM are unlocked. These registers can be written by CPU and SPT
30 WR32_ACC_ CTRL	Work register 32 access control 0: SPT has write access to WR_R32_RE and WR_R32_IM 1: CPU has write access to WR_R32_RE and WR_R32_IM
31 WR32_LCK	Work register 32 lock 1: Writes to WR_R32_RE and WR_R32_IM are locked. These registers written by either CPU or SPT as defined by WR32_ACC_CTRL bit. 0: Writes to WR_R32_RE and WR_R32_IM are unlocked. These registers can be written by CPU and SPT

45.8.95 Work Register R0 Real (SPT_WR_R0_RE)

Address: 0h base + 250h offset = 250h

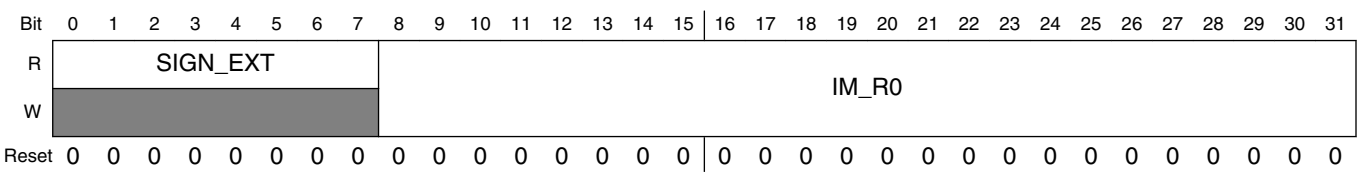


SPT_WR_R0_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R0	Real part of Work Register R0

45.8.96 Work Register R0 Imaginary (SPT_WR_R0_IM)

Address: 0h base + 254h offset = 254h

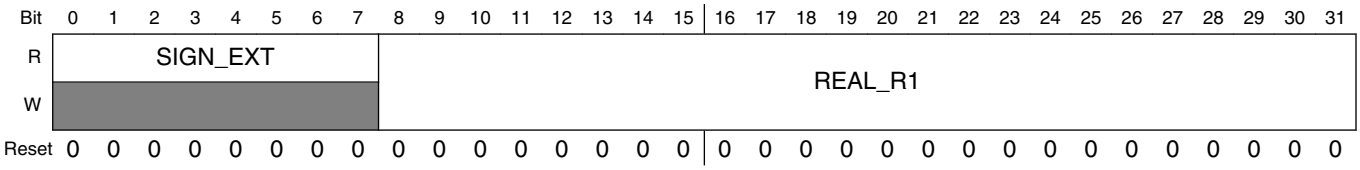


SPT_WR_R0_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R0	Imaginary part of Work Register R0

45.8.97 Work Register R1 Real (SPT_WR_R1_RE)

Address: 0h base + 258h offset = 258h

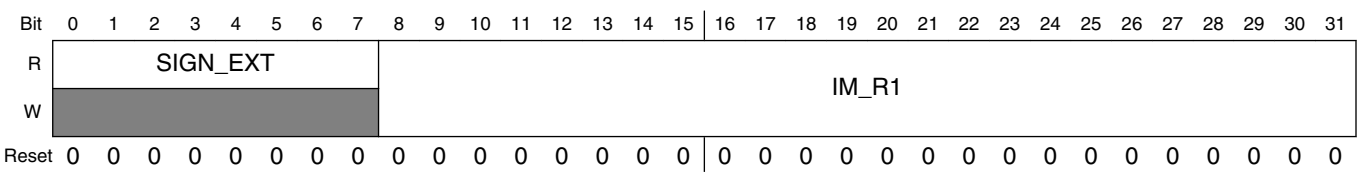


SPT_WR_R1_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R1	Real part of Work Register R1

45.8.98 Work Register R1 Imaginary (SPT_WR_R1_IM)

Address: 0h base + 25Ch offset = 25Ch

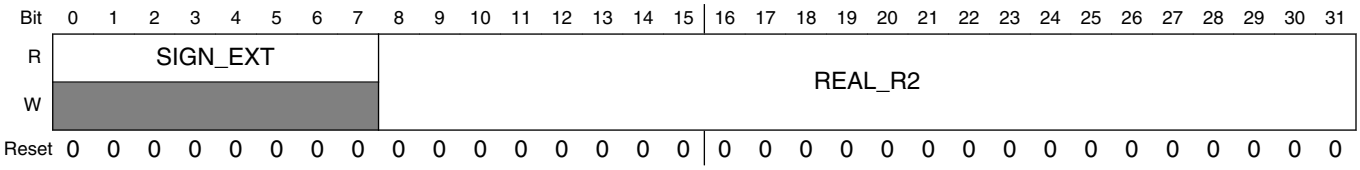


SPT_WR_R1_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R1	Imaginary part of Work Register R1

45.8.99 Work Register R0 Real (SPT_WR_R2_RE)

Address: 0h base + 260h offset = 260h

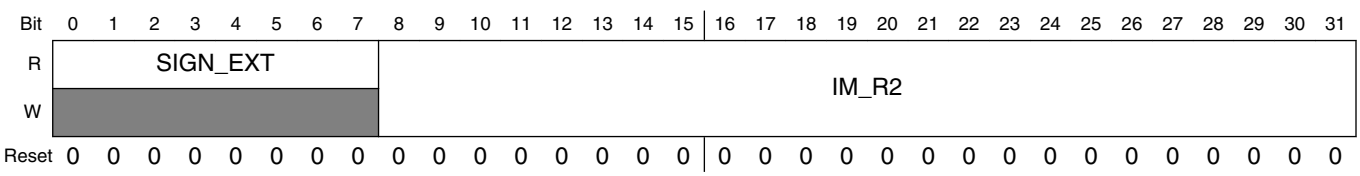


SPT_WR_R2_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R2	Real part of Work Register R2

45.8.100 Work Register R2 Imaginary (SPT_WR_R2_IM)

Address: 0h base + 264h offset = 264h

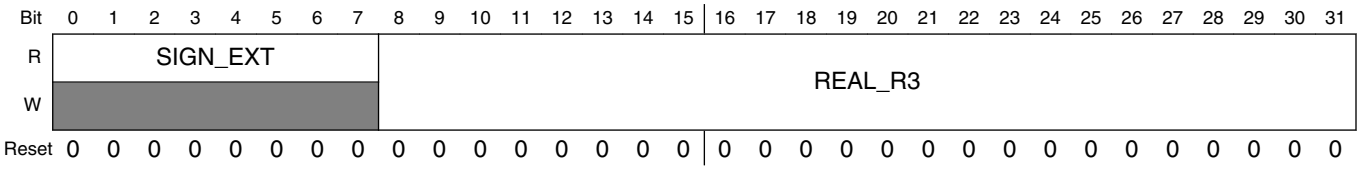


SPT_WR_R2_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R2	Imaginary part of Work Register R2

45.8.101 Work Register R3 Real (SPT_WR_R3_RE)

Address: 0h base + 268h offset = 268h

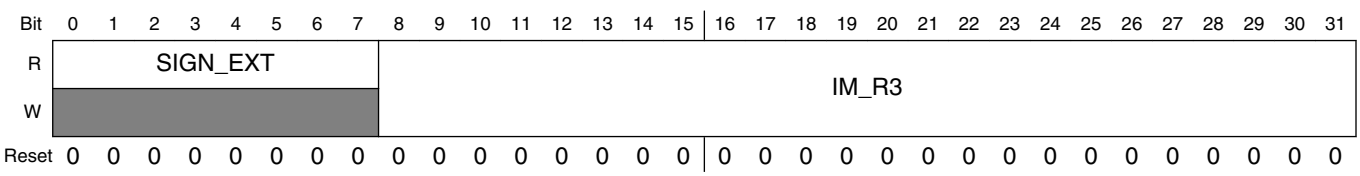


SPT_WR_R3_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R3	Real part of Work Register R3

45.8.102 Work Register R3 Imaginary (SPT_WR_R3_IM)

Address: 0h base + 26Ch offset = 26Ch

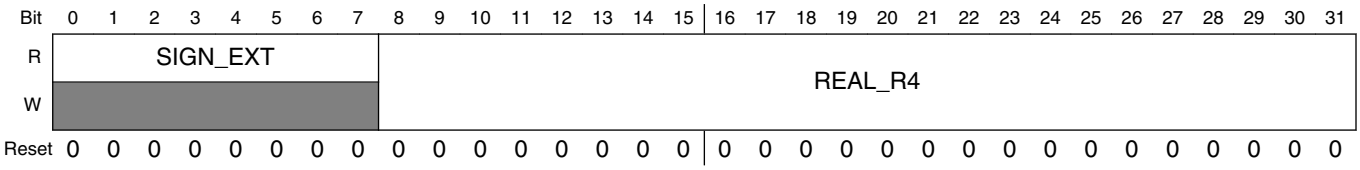


SPT_WR_R3_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R3	Imaginary part of Work Register R3

45.8.103 Work Register R4 Real (SPT_WR_R4_RE)

Address: 0h base + 270h offset = 270h

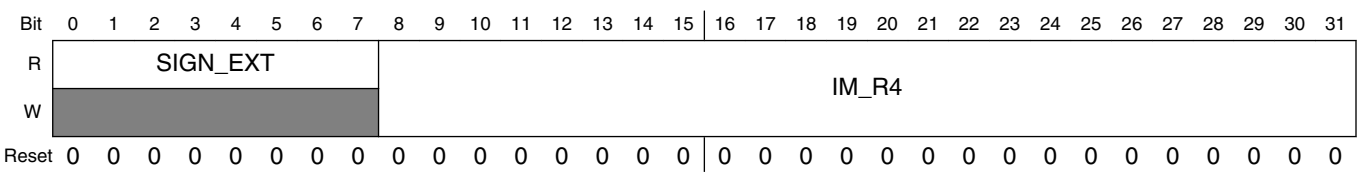


SPT_WR_R4_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R4	Real part of Work Register R4

45.8.104 Work Register R4 Imaginary (SPT_WR_R4_IM)

Address: 0h base + 274h offset = 274h

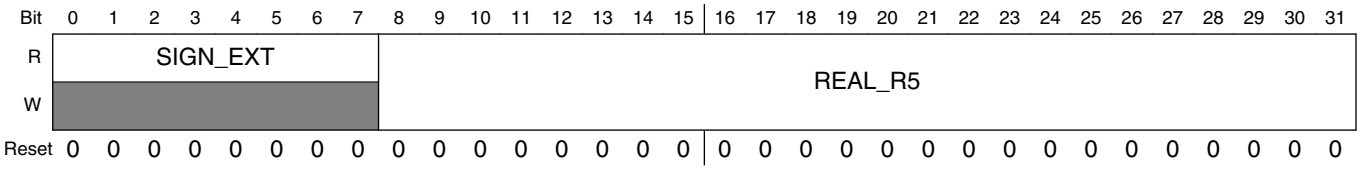


SPT_WR_R4_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R4	Imaginary part of Work Register R4

45.8.105 Work Register R5 Real (SPT_WR_R5_RE)

Address: 0h base + 278h offset = 278h

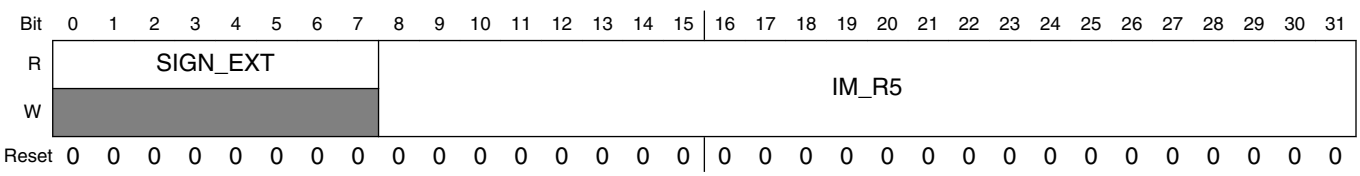


SPT_WR_R5_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R5	Real part of Work Register R5

45.8.106 Work Register R5 Imaginary (SPT_WR_R5_IM)

Address: 0h base + 27Ch offset = 27Ch

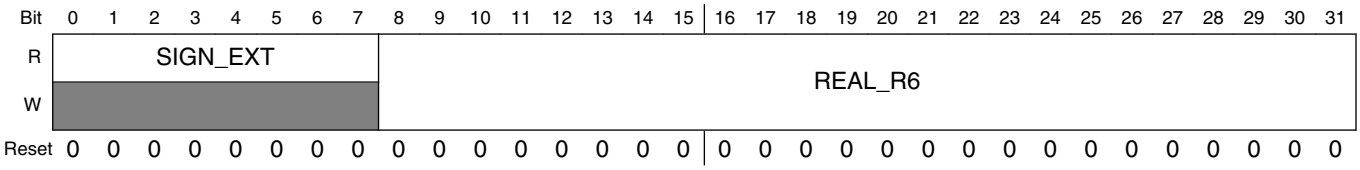


SPT_WR_R5_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R5	Imaginary part of Work Register R5

45.8.107 Work Register R6 Real (SPT_WR_R6_RE)

Address: 0h base + 280h offset = 280h

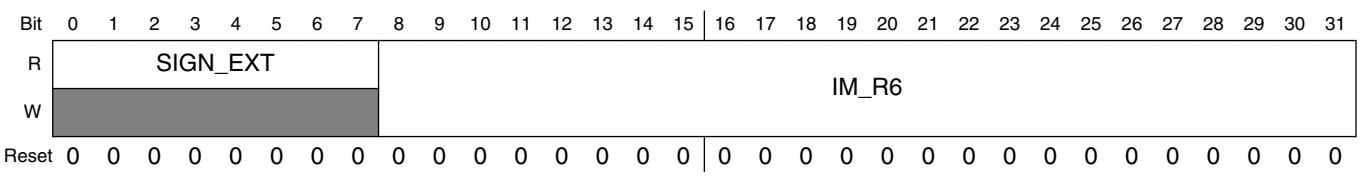


SPT_WR_R6_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R6	Real part of Work Register R6

45.8.108 Work Register R6 Imaginary (SPT_WR_R6_IM)

Address: 0h base + 284h offset = 284h

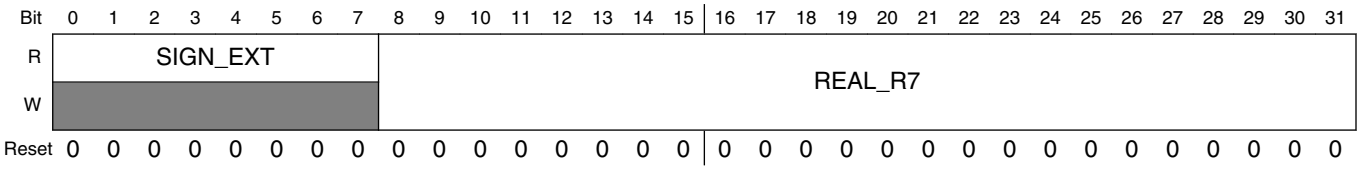


SPT_WR_R6_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R6	Imaginary part of Work Register R6

45.8.109 Work Register R7 Real (SPT_WR_R7_RE)

Address: 0h base + 288h offset = 288h

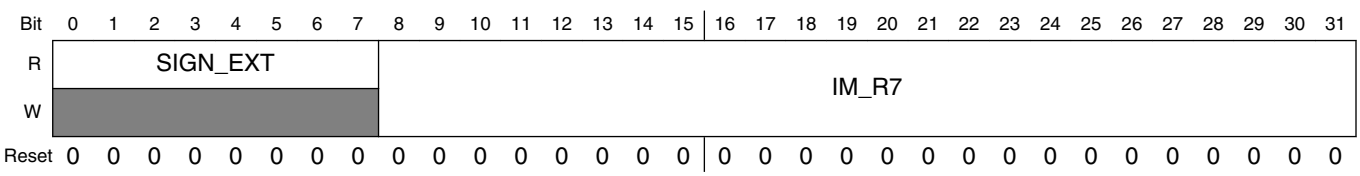


SPT_WR_R7_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R7	Real part of Work Register R7

45.8.110 Work Register R7 Imaginary (SPT_WR_R7_IM)

Address: 0h base + 28Ch offset = 28Ch

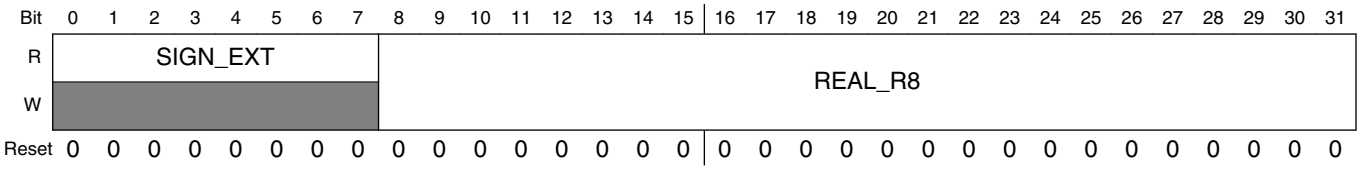


SPT_WR_R7_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R7	Imaginary part of Work Register R7

45.8.111 Work Register R8 Real (SPT_WR_R8_RE)

Address: 0h base + 290h offset = 290h

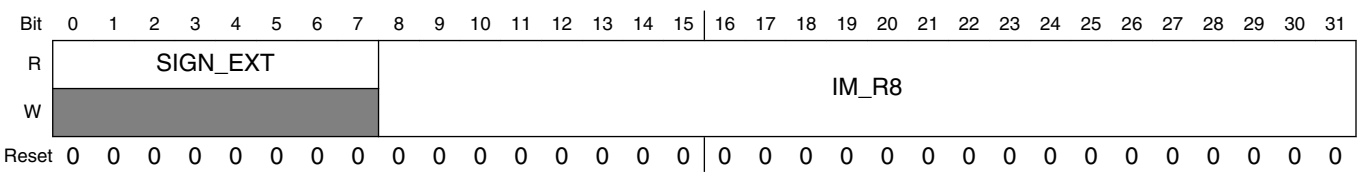


SPT_WR_R8_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R8	Real part of Work Register R8

45.8.112 Work Register R8 Imaginary (SPT_WR_R8_IM)

Address: 0h base + 294h offset = 294h

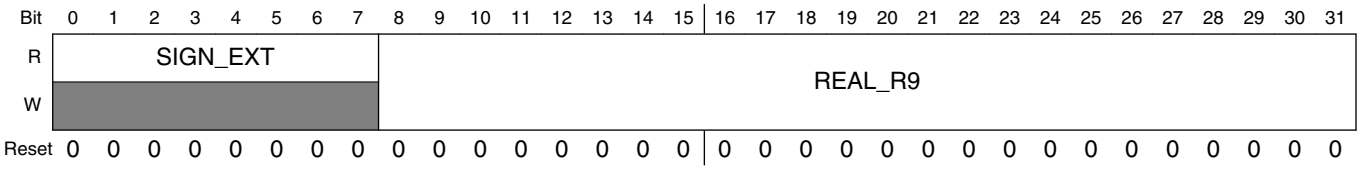


SPT_WR_R8_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R8	Imaginary part of Work Register R8

45.8.113 Work Register R9 Real (SPT_WR_R9_RE)

Address: 0h base + 298h offset = 298h

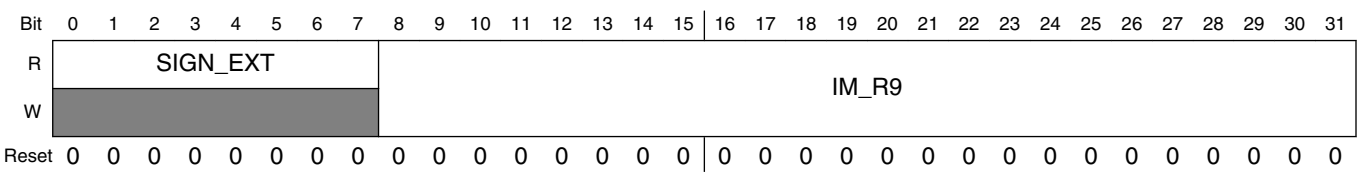


SPT_WR_R9_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R9	Real part of Work Register R9

45.8.114 Work Register R9 Imaginary (SPT_WR_R9_IM)

Address: 0h base + 29Ch offset = 29Ch

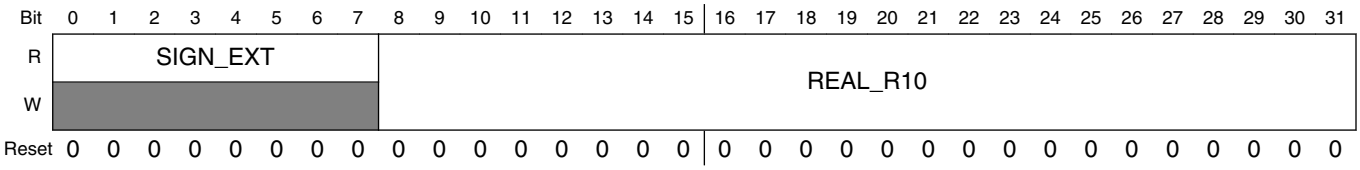


SPT_WR_R9_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R9	Imaginary part of Work Register R9

45.8.115 Work Register R10 Real (SPT_WR_R10_RE)

Address: 0h base + 2A0h offset = 2A0h

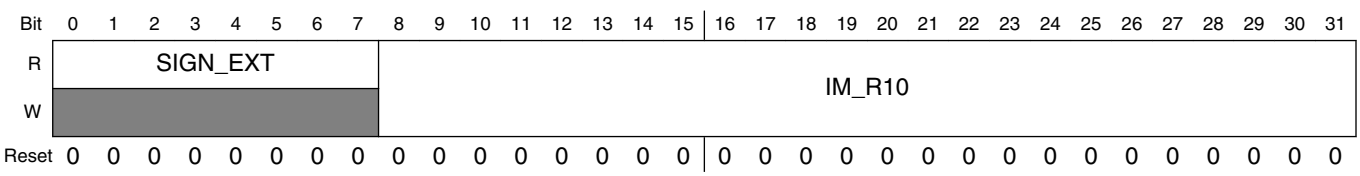


SPT_WR_R10_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R10	Real part of Work Register R10

45.8.116 Work Register R10 Imaginary (SPT_WR_R10_IM)

Address: 0h base + 2A4h offset = 2A4h

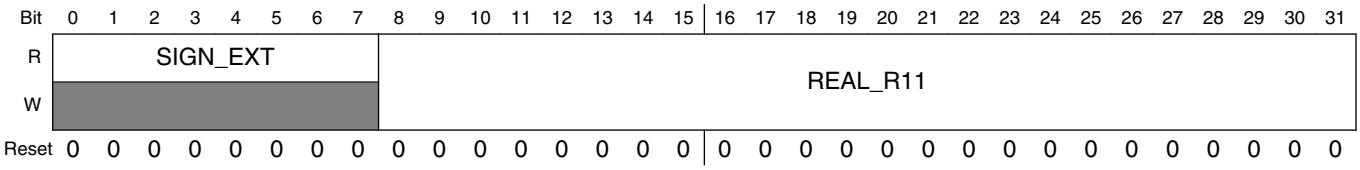


SPT_WR_R10_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R10	Imaginary part of Work Register R10

45.8.117 Work Register R11 Real (SPT_WR_R11_RE)

Address: 0h base + 2A8h offset = 2A8h

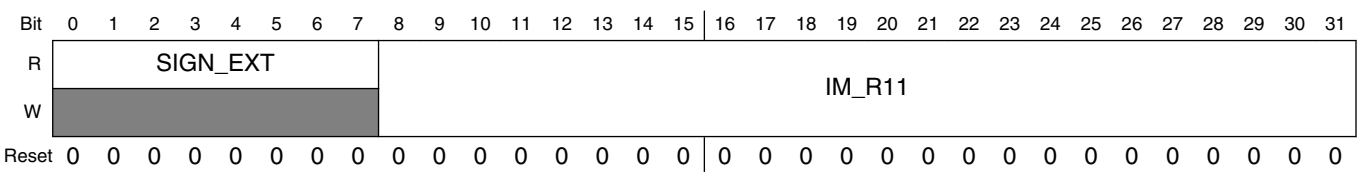


SPT_WR_R11_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R11	Real part of Work Register R11

45.8.118 Work Register R11 Imaginary (SPT_WR_R11_IM)

Address: 0h base + 2ACh offset = 2ACh

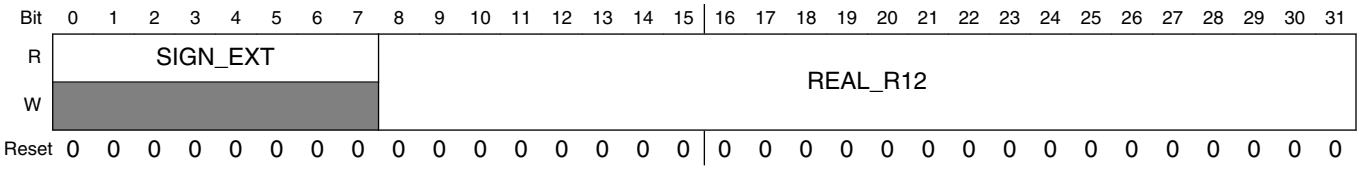


SPT_WR_R11_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R11	Imaginary part of Work Register R11

45.8.119 Work Register R12 Real (SPT_WR_R12_RE)

Address: 0h base + 2B0h offset = 2B0h

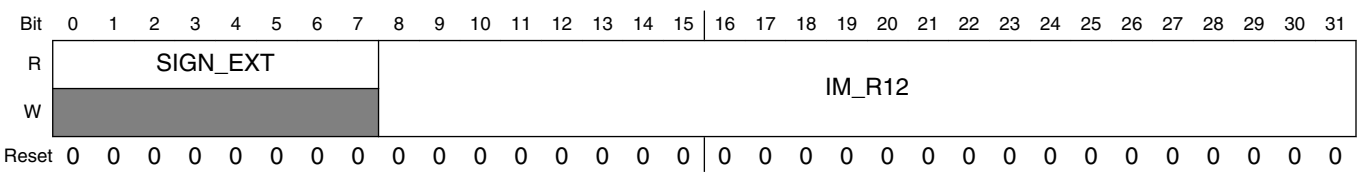


SPT_WR_R12_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R12	Real part of Work Register R12

45.8.120 Work Register R12 Imaginary (SPT_WR_R12_IM)

Address: 0h base + 2B4h offset = 2B4h

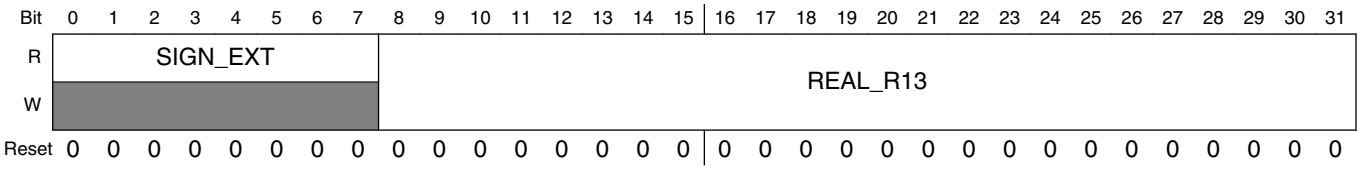


SPT_WR_R12_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R12	Imaginary part of Work Register R12

45.8.121 Work Register R13 Real (SPT_WR_R13_RE)

Address: 0h base + 2B8h offset = 2B8h

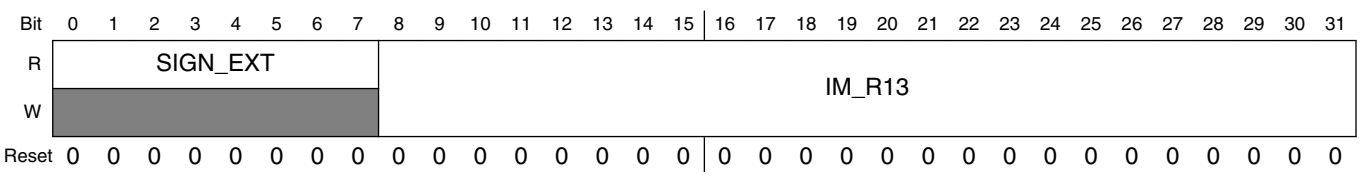


SPT_WR_R13_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R13	Real part of Work Register R13

45.8.122 Work Register R13 Imaginary (SPT_WR_R13_IM)

Address: 0h base + 2BCh offset = 2BCh

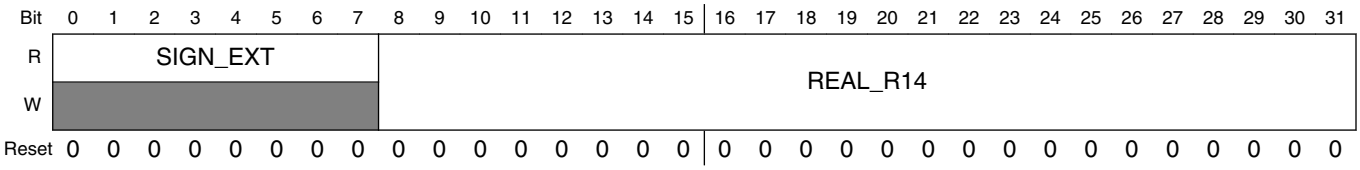


SPT_WR_R13_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R13	Imaginary part of Work Register R13

45.8.123 Work Register R14 Real (SPT_WR_R14_RE)

Address: 0h base + 2C0h offset = 2C0h

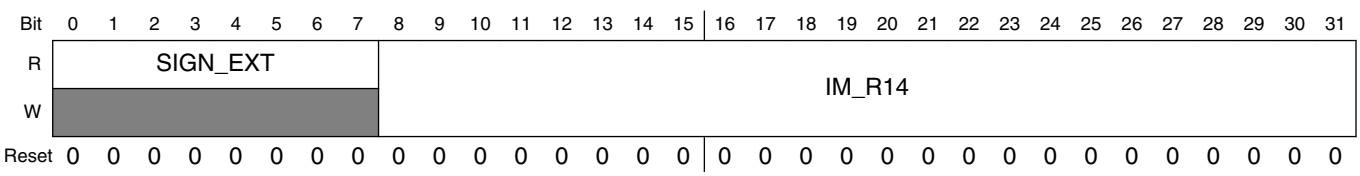


SPT_WR_R14_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R14	Real part of Work Register R14

45.8.124 Work Register R14 Imaginary (SPT_WR_R14_IM)

Address: 0h base + 2C4h offset = 2C4h

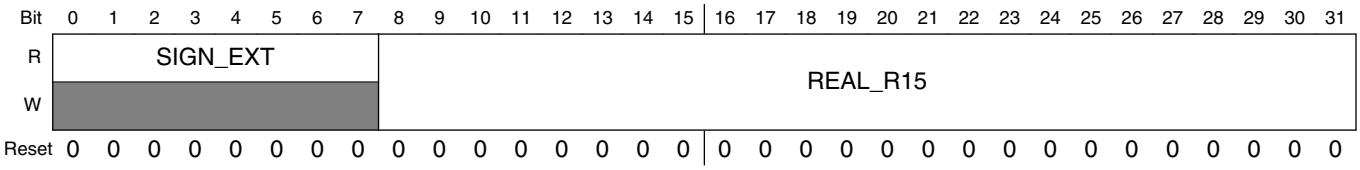


SPT_WR_R14_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R14	Imaginary part of Work Register R14

45.8.125 Work Register R15 Real (SPT_WR_R15_RE)

Address: 0h base + 2C8h offset = 2C8h

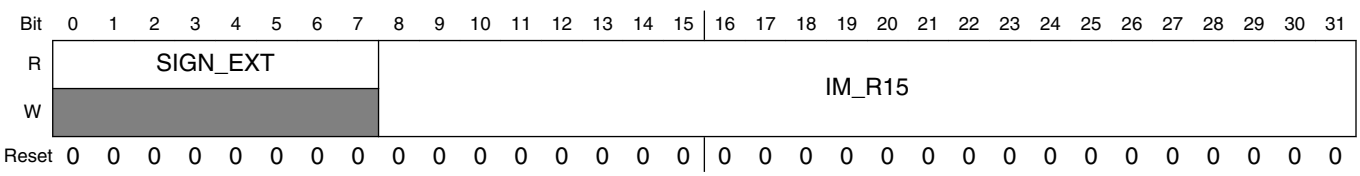


SPT_WR_R15_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R15	Real part of Work Register R15

45.8.126 Work Register R15 Imaginary (SPT_WR_R15_IM)

Address: 0h base + 2CCh offset = 2CCh

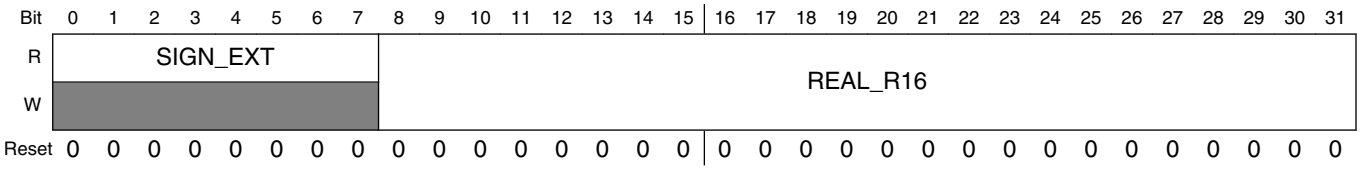


SPT_WR_R15_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R15	Imaginary part of Work Register R15

45.8.127 Work Register R16 Real (SPT_WR_R16_RE)

Address: 0h base + 2D0h offset = 2D0h

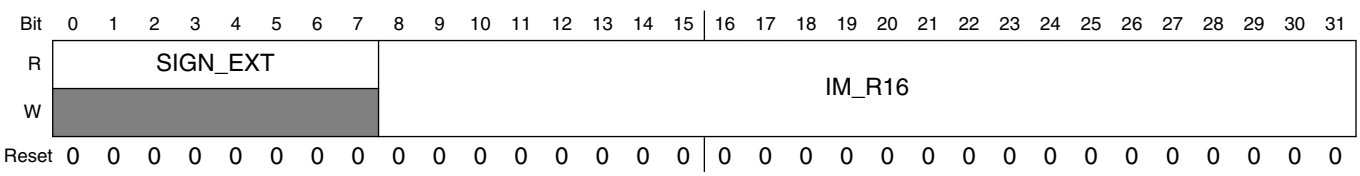


SPT_WR_R16_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R16	Real part of Work Register R16

45.8.128 Work Register R16 Imaginary (SPT_WR_R16_IM)

Address: 0h base + 2D4h offset = 2D4h

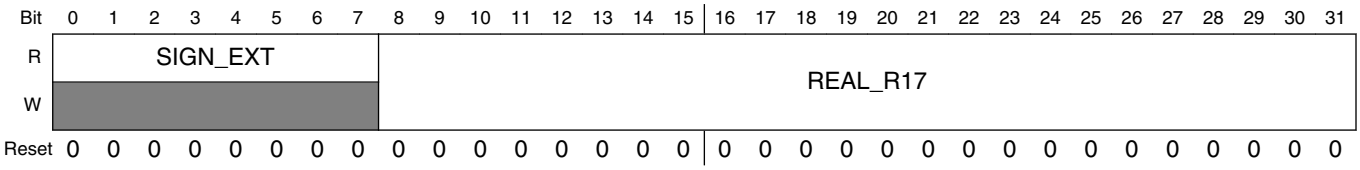


SPT_WR_R16_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R16	Imaginary part of Work Register R16

45.8.129 Work Register R17 Real (SPT_WR_R17_RE)

Address: 0h base + 2D8h offset = 2D8h

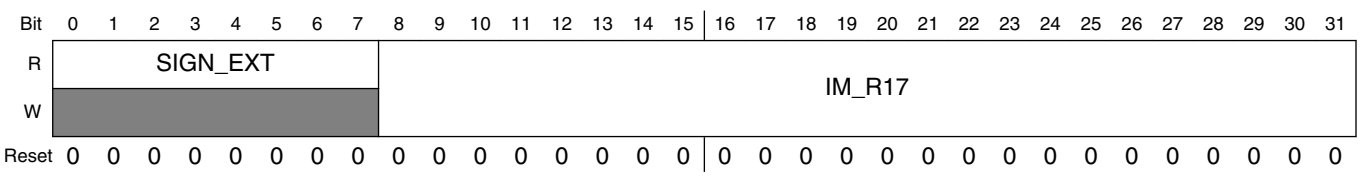


SPT_WR_R17_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R17	Real part of Work Register R17

45.8.130 Work Register R17 Imaginary (SPT_WR_R17_IM)

Address: 0h base + 2DCh offset = 2DCh

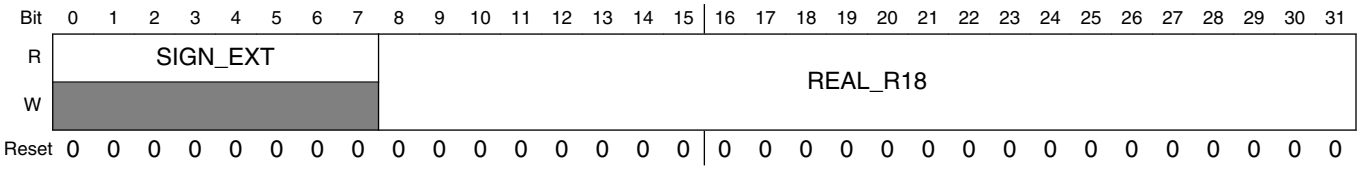


SPT_WR_R17_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R17	Imaginary part of Work Register R17

45.8.131 Work Register R18 Real (SPT_WR_R18_RE)

Address: 0h base + 2E0h offset = 2E0h

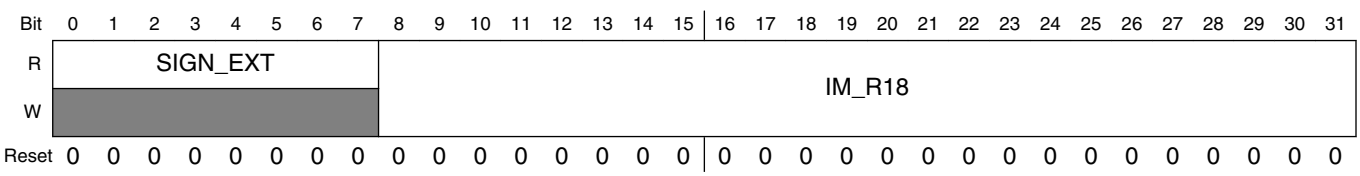


SPT_WR_R18_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R18	Real part of Work Register R18

45.8.132 Work Register R18 Imaginary (SPT_WR_R18_IM)

Address: 0h base + 2E4h offset = 2E4h

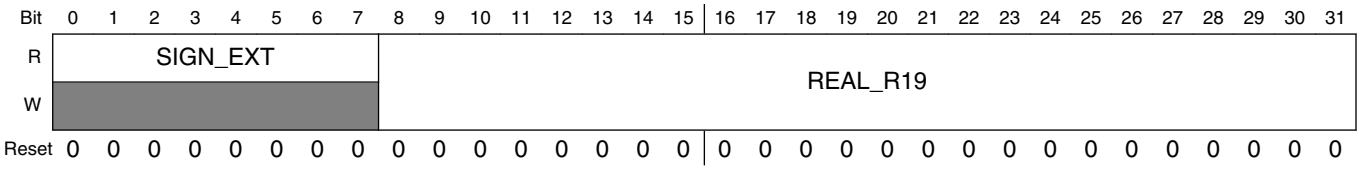


SPT_WR_R18_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R18	Imaginary part of Work Register R18

45.8.133 Work Register R19 Real (SPT_WR_R19_RE)

Address: 0h base + 2E8h offset = 2E8h

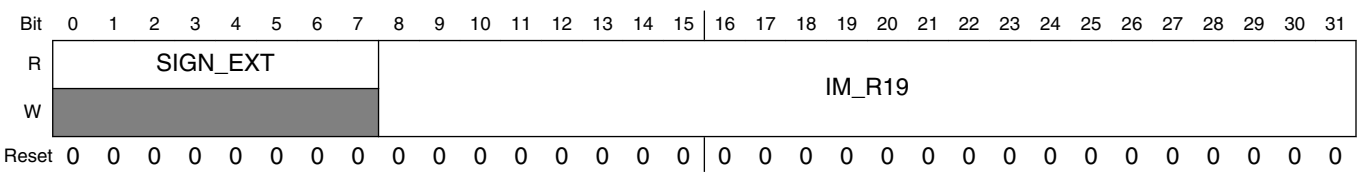


SPT_WR_R19_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R19	Real part of Work Register R19

45.8.134 Work Register R19 Imaginary (SPT_WR_R19_IM)

Address: 0h base + 2ECh offset = 2ECh

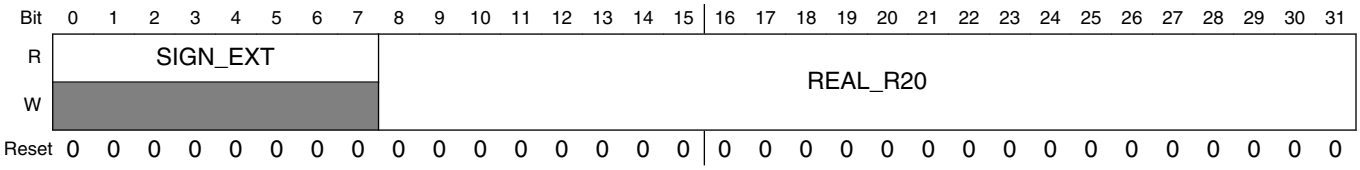


SPT_WR_R19_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R19	Imaginary part of Work Register R19

45.8.135 Work Register R20 Real (SPT_WR_R20_RE)

Address: 0h base + 2F0h offset = 2F0h

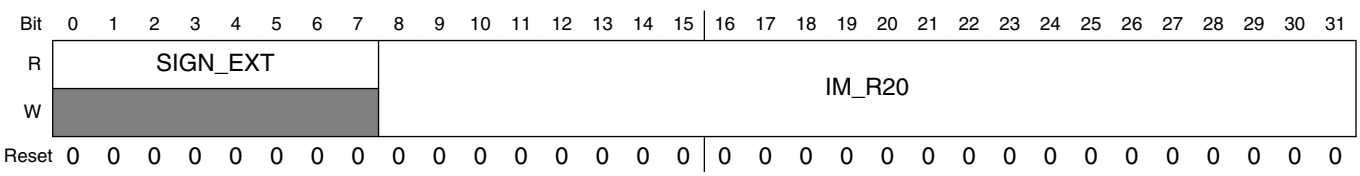


SPT_WR_R20_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R20	Real part of Work Register R20

45.8.136 Work Register R20 Imaginary (SPT_WR_R20_IM)

Address: 0h base + 2F4h offset = 2F4h

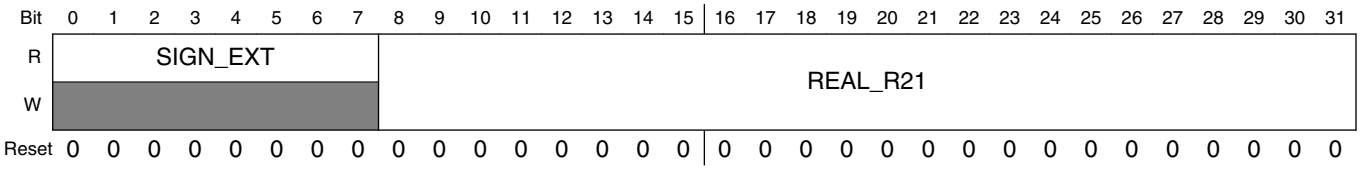


SPT_WR_R20_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R20	Imaginary part of Work Register R20

45.8.137 Work Register R21 Real (SPT_WR_R21_RE)

Address: 0h base + 2F8h offset = 2F8h

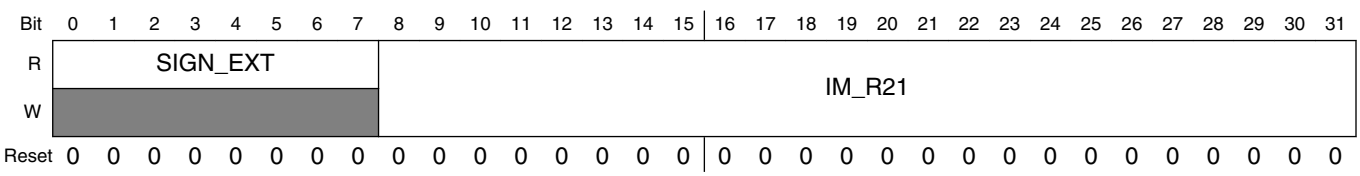


SPT_WR_R21_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R21	Real part of Work Register R21

45.8.138 Work Register R21 Imaginary (SPT_WR_R21_IM)

Address: 0h base + 2FCh offset = 2FCh

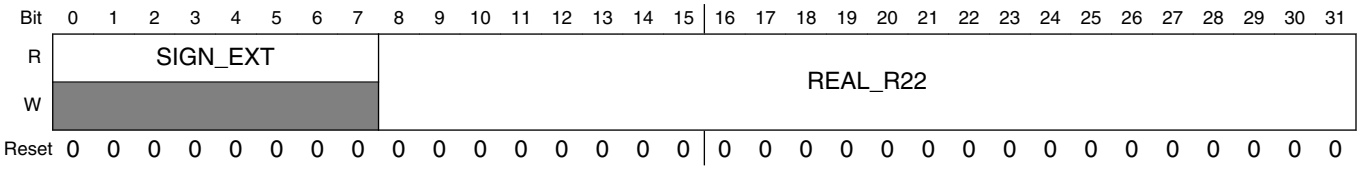


SPT_WR_R21_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R21	Imaginary part of Work Register R21

45.8.139 Work Register R22 Real (SPT_WR_R22_RE)

Address: 0h base + 300h offset = 300h

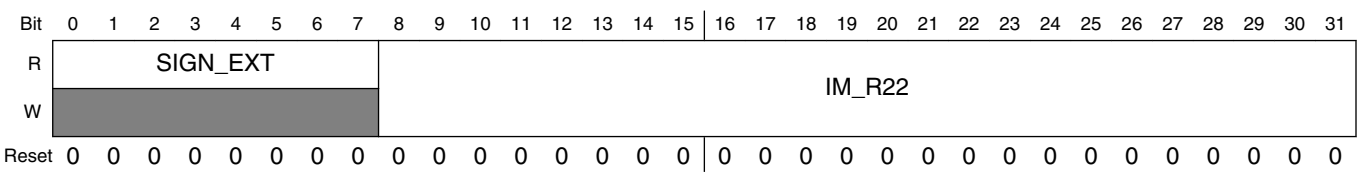


SPT_WR_R22_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R22	Real part of Work Register R22

45.8.140 Work Register R22 Imaginary (SPT_WR_R22_IM)

Address: 0h base + 304h offset = 304h

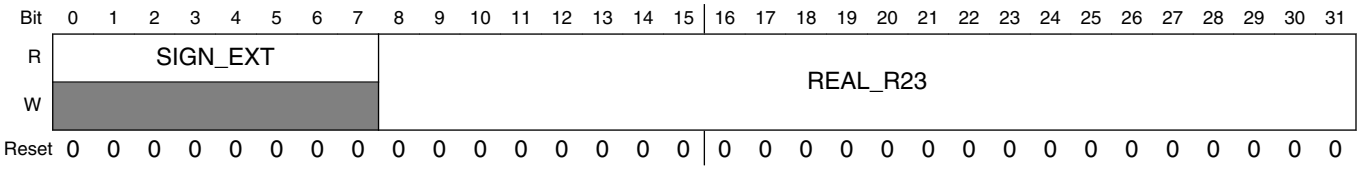


SPT_WR_R22_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R22	Imaginary part of Work Register R22

45.8.141 Work Register R23 Real (SPT_WR_R23_RE)

Address: 0h base + 308h offset = 308h

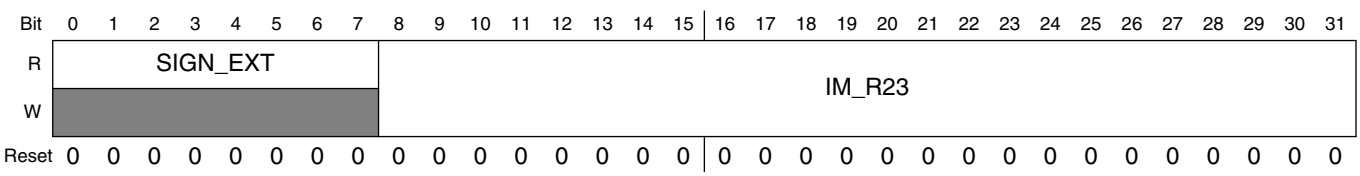


SPT_WR_R23_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R23	Real part of Work Register R23

45.8.142 Work Register R23 Imaginary (SPT_WR_R23_IM)

Address: 0h base + 30Ch offset = 30Ch

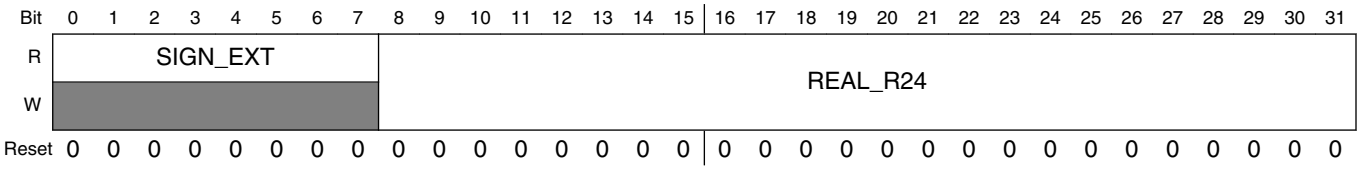


SPT_WR_R23_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R23	Imaginary part of Work Register R23

45.8.143 Work Register R24 Real (SPT_WR_R24_RE)

Address: 0h base + 310h offset = 310h

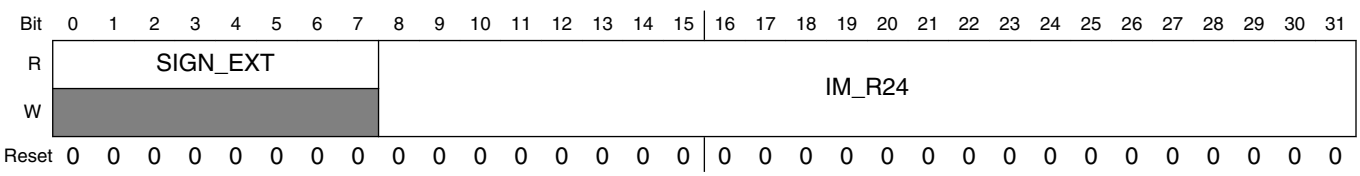


SPT_WR_R24_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R24	Real part of Work Register R24

45.8.144 Work Register R24 Imaginary (SPT_WR_R24_IM)

Address: 0h base + 314h offset = 314h

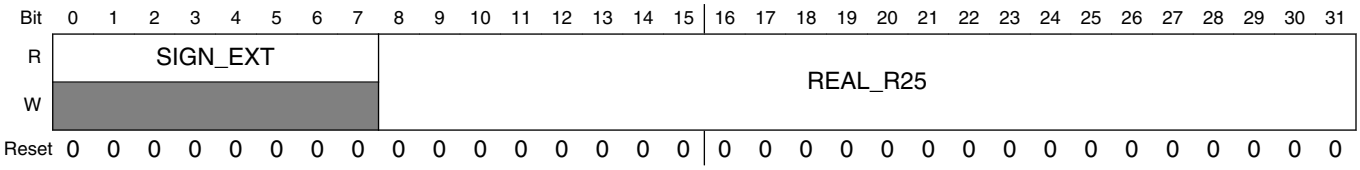


SPT_WR_R24_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R24	Imaginary part of Work Register R24

45.8.145 Work Register R25 Real (SPT_WR_R25_RE)

Address: 0h base + 318h offset = 318h

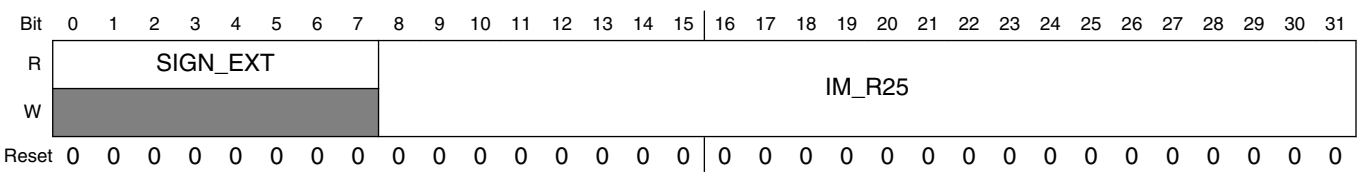


SPT_WR_R25_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R25	Real part of Work Register R25

45.8.146 Work Register R25 Imaginary (SPT_WR_R25_IM)

Address: 0h base + 31Ch offset = 31Ch

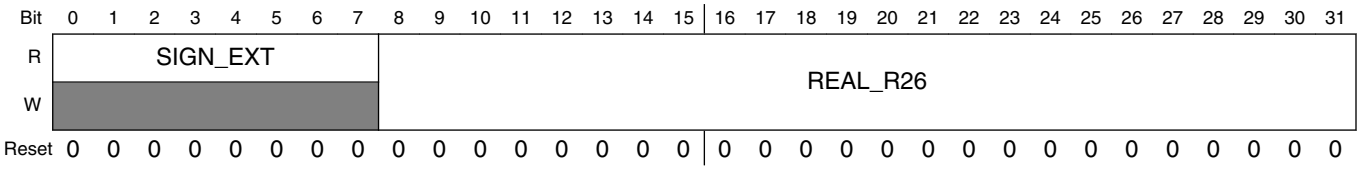


SPT_WR_R25_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R25	Imaginary part of Work Register R25

45.8.147 Work Register R26 Real (SPT_WR_R26_RE)

Address: 0h base + 320h offset = 320h

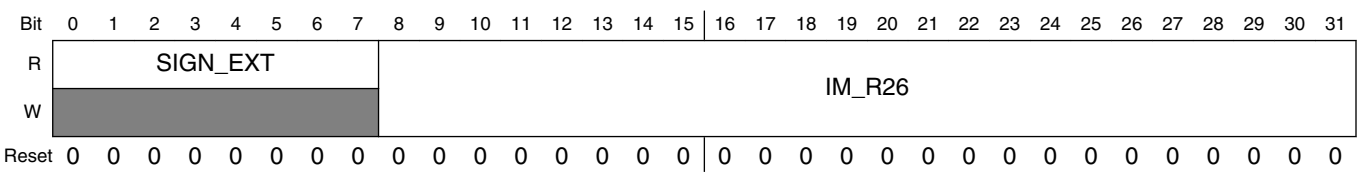


SPT_WR_R26_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R26	Real part of Work Register R26

45.8.148 Work Register R26 Imaginary (SPT_WR_R26_IM)

Address: 0h base + 324h offset = 324h

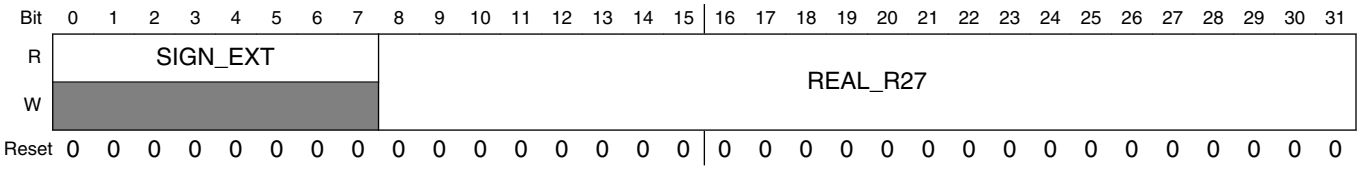


SPT_WR_R26_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R26	Imaginary part of Work Register R26

45.8.149 Work Register R27 Real (SPT_WR_R27_RE)

Address: 0h base + 328h offset = 328h

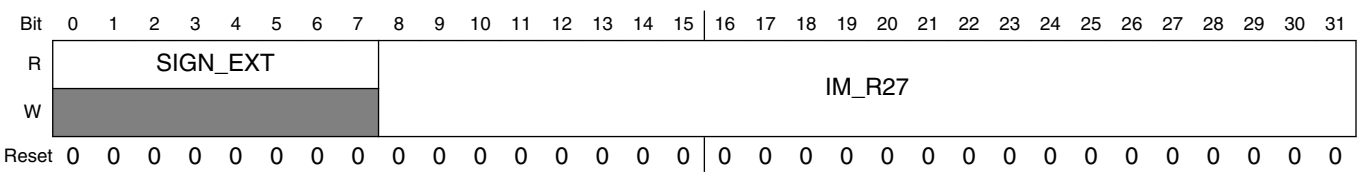


SPT_WR_R27_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R27	Real part of Work Register R27

45.8.150 Work Register R27 Imaginary (SPT_WR_R27_IM)

Address: 0h base + 32Ch offset = 32Ch

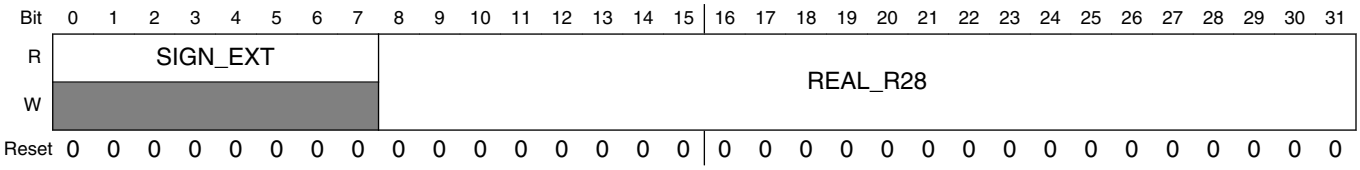


SPT_WR_R27_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R27	Imaginary part of Work Register R27

45.8.151 Work Register R28 Real (SPT_WR_R28_RE)

Address: 0h base + 330h offset = 330h

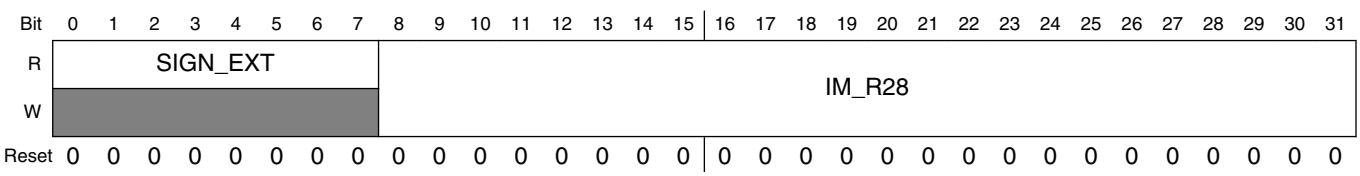


SPT_WR_R28_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R28	Real part of Work Register R28

45.8.152 Work Register R28 Imaginary (SPT_WR_R28_IM)

Address: 0h base + 334h offset = 334h

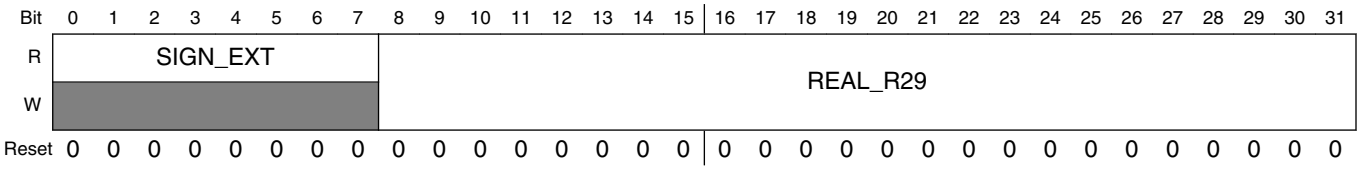


SPT_WR_R28_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R28	Imaginary part of Work Register R28

45.8.153 Work Register R29 Real (SPT_WR_R29_RE)

Address: 0h base + 338h offset = 338h

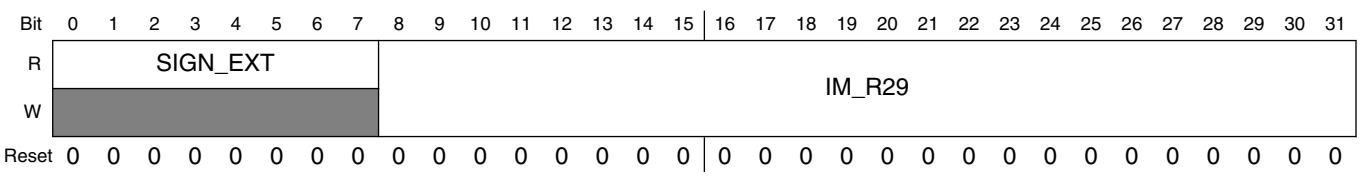


SPT_WR_R29_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R29	Real part of Work Register R29

45.8.154 Work Register R29 Imaginary (SPT_WR_R29_IM)

Address: 0h base + 33Ch offset = 33Ch

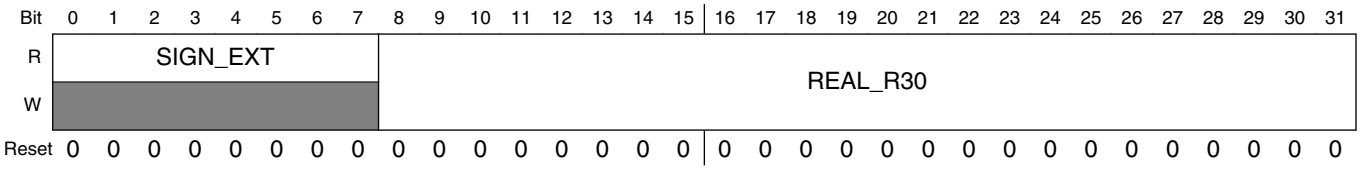


SPT_WR_R29_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R29	Imaginary part of Work Register R29

45.8.155 Work Register R30 Real (SPT_WR_R30_RE)

Address: 0h base + 340h offset = 340h

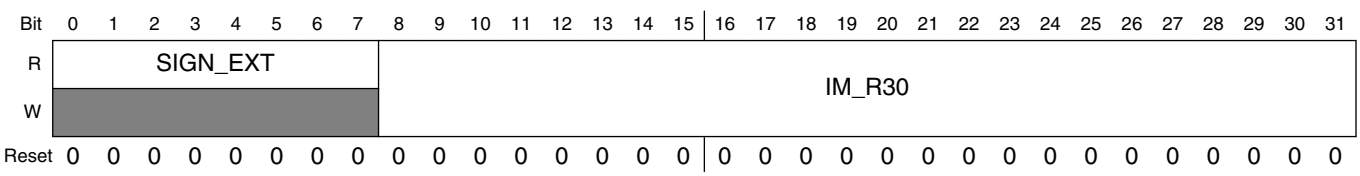


SPT_WR_R30_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R30	Real part of Work Register R30

45.8.156 Work Register R30 Imaginary (SPT_WR_R30_IM)

Address: 0h base + 344h offset = 344h

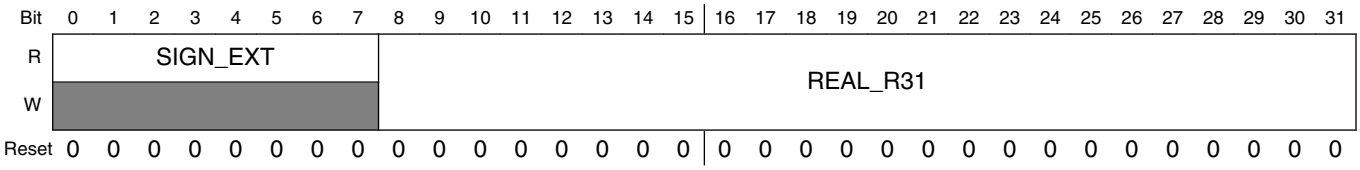


SPT_WR_R30_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R30	Imaginary part of Work Register R30

45.8.157 Work Register R31 Real (SPT_WR_R31_RE)

Address: 0h base + 348h offset = 348h

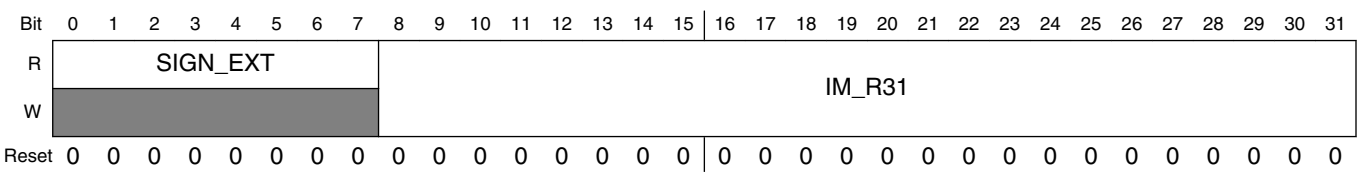


SPT_WR_R31_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R31	Real part of Work Register R31

45.8.158 Work Register R31 Imaginary (SPT_WR_R31_IM)

Address: 0h base + 34Ch offset = 34Ch

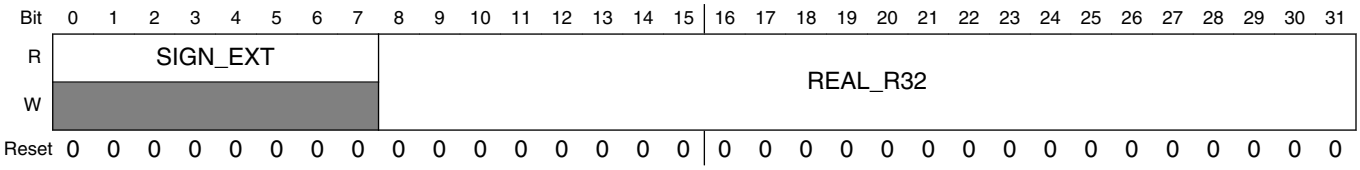


SPT_WR_R31_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R31	Imaginary part of Work Register R31

45.8.159 Work Register R32 Real (SPT_WR_R32_RE)

Address: 0h base + 350h offset = 350h

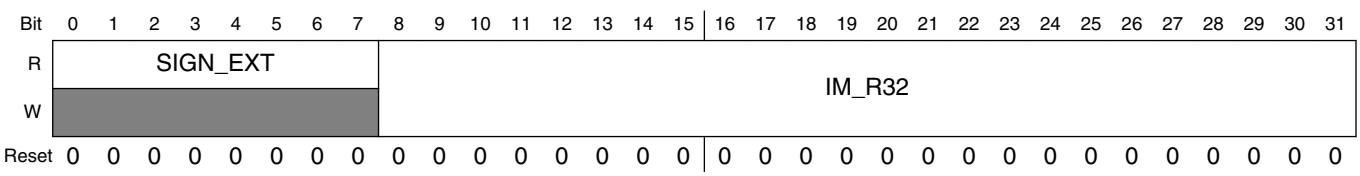


SPT_WR_R32_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R32	Real part of Work Register R32

45.8.160 Work Register R32 Imaginary (SPT_WR_R32_IM)

Address: 0h base + 354h offset = 354h

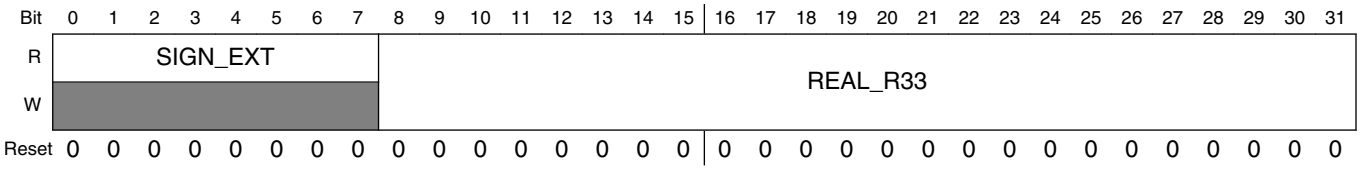


SPT_WR_R32_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R32	Imaginary part of Work Register R32

45.8.161 Work Register R33 Real (SPT_WR_R33_RE)

Address: 0h base + 358h offset = 358h

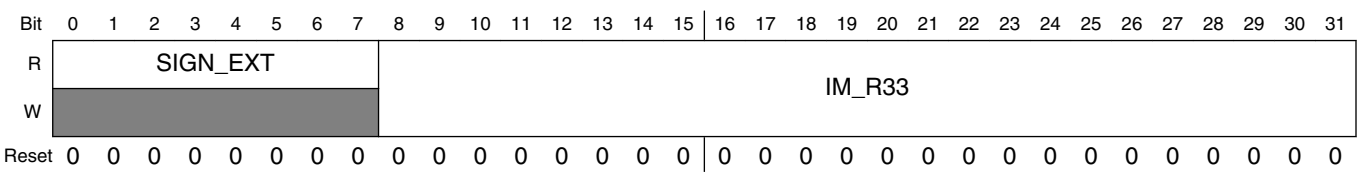


SPT_WR_R33_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R33	Real part of Work Register R33

45.8.162 Work Register R33 Imaginary (SPT_WR_R33_IM)

Address: 0h base + 35Ch offset = 35Ch

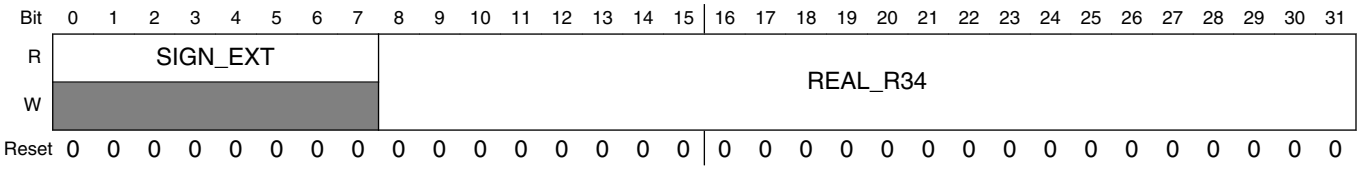


SPT_WR_R33_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R33	Imaginary part of Work Register R33

45.8.163 Work Register R34 Real (SPT_WR_R34_RE)

Address: 0h base + 360h offset = 360h

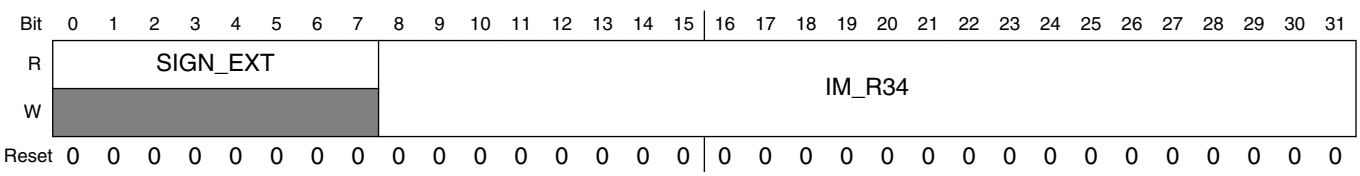


SPT_WR_R34_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R34	Real part of Work Register R34

45.8.164 Work Register R34 Imaginary (SPT_WR_R34_IM)

Address: 0h base + 364h offset = 364h

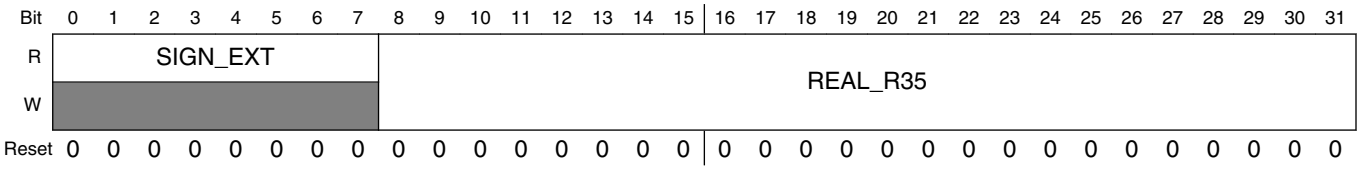


SPT_WR_R34_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R34	Imaginary part of Work Register R34

45.8.165 Work Register R35 Real (SPT_WR_R35_RE)

Address: 0h base + 368h offset = 368h

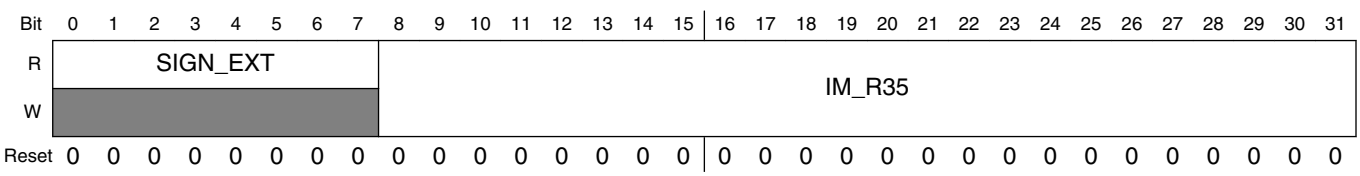


SPT_WR_R35_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R35	Real part of Work Register R35

45.8.166 Work Register R35 Imaginary (SPT_WR_R35_IM)

Address: 0h base + 36Ch offset = 36Ch

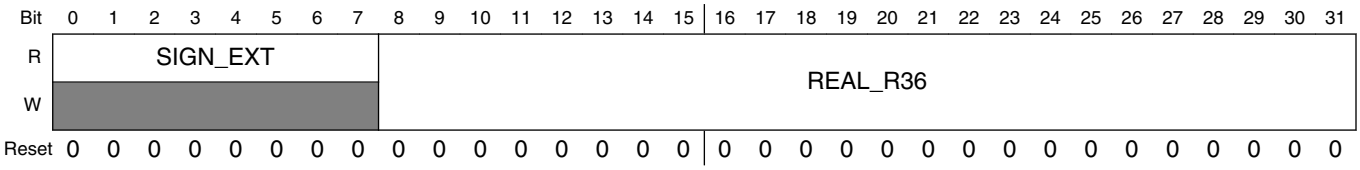


SPT_WR_R35_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R35	Imaginary part of Work Register R35

45.8.167 Work Register R36 Real (SPT_WR_R36_RE)

Address: 0h base + 370h offset = 370h

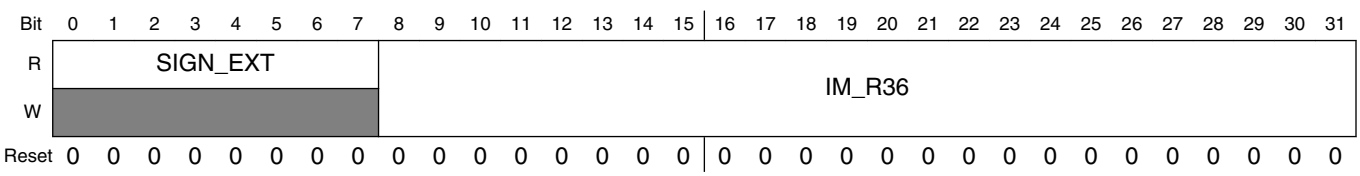


SPT_WR_R36_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R36	Real part of Work Register R36

45.8.168 Work Register R36 Imaginary (SPT_WR_R36_IM)

Address: 0h base + 374h offset = 374h

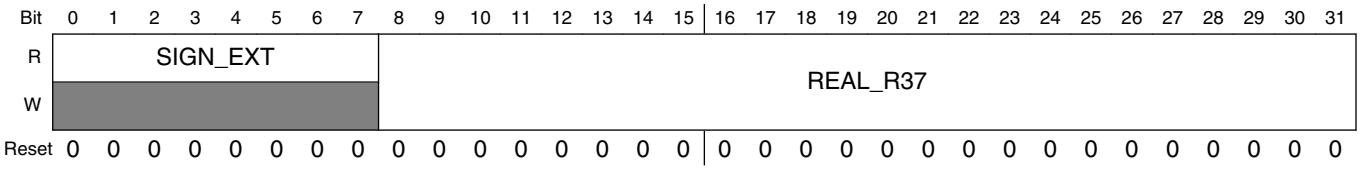


SPT_WR_R36_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R36	Imaginary part of Work Register R36

45.8.169 Work Register R37 Real (SPT_WR_R37_RE)

Address: 0h base + 378h offset = 378h

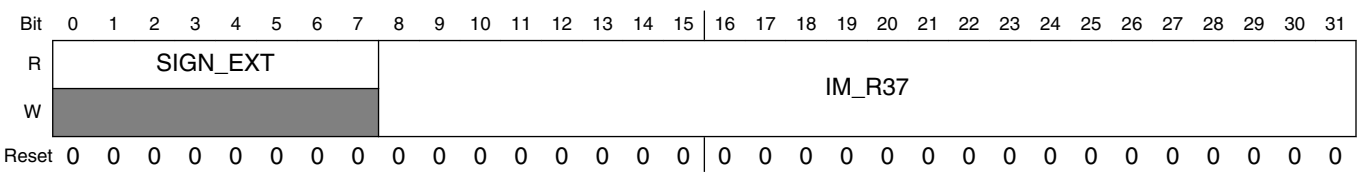


SPT_WR_R37_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R37	Real part of Work Register R37

45.8.170 Work Register R37 Imaginary (SPT_WR_R37_IM)

Address: 0h base + 37Ch offset = 37Ch

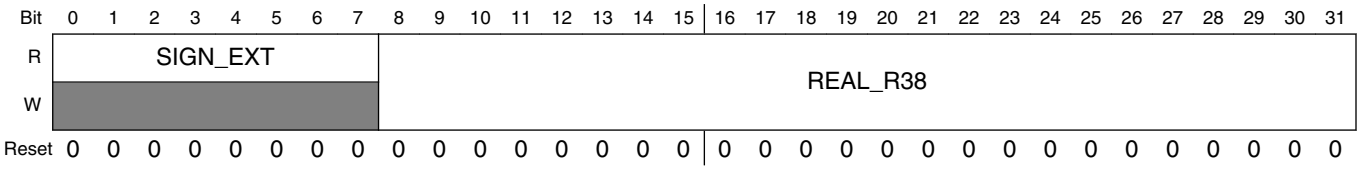


SPT_WR_R37_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R37	Imaginary part of Work Register R37

45.8.171 Work Register R38 Real (SPT_WR_R38_RE)

Address: 0h base + 380h offset = 380h

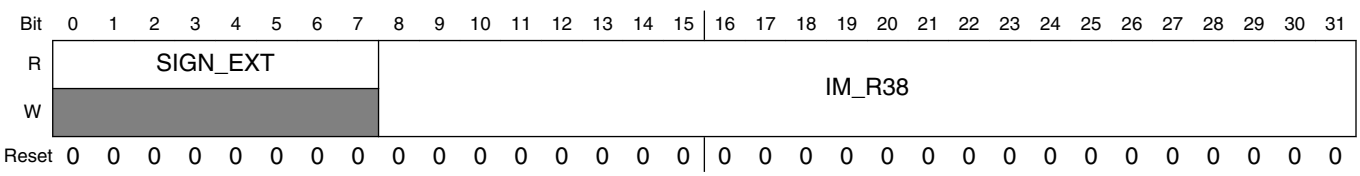


SPT_WR_R38_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R38	Real part of Work Register R38

45.8.172 Work Register R38 Imaginary (SPT_WR_R38_IM)

Address: 0h base + 384h offset = 384h

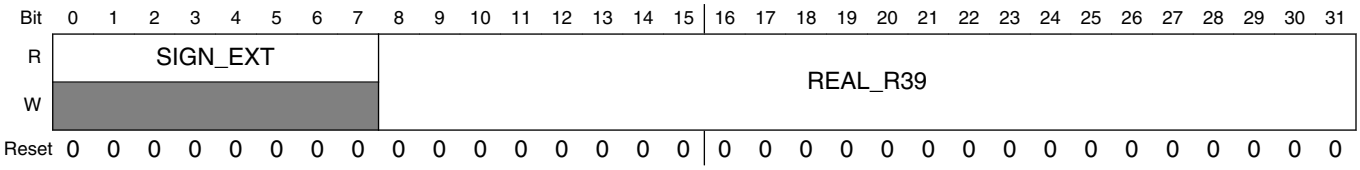


SPT_WR_R38_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R38	Imaginary part of Work Register R38

45.8.173 Work Register R39 Real (SPT_WR_R39_RE)

Address: 0h base + 388h offset = 388h

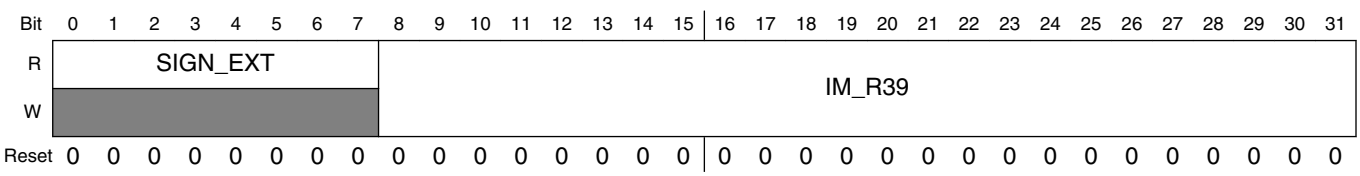


SPT_WR_R39_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R39	Real part of Work Register R39

45.8.174 Work Register R39 Imaginary (SPT_WR_R39_IM)

Address: 0h base + 38Ch offset = 38Ch

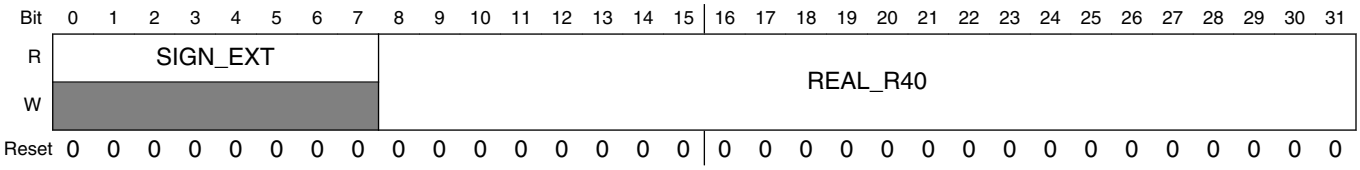


SPT_WR_R39_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R39	Imaginary part of Work Register R39

45.8.175 Work Register R40 Real (SPT_WR_R40_RE)

Address: 0h base + 390h offset = 390h

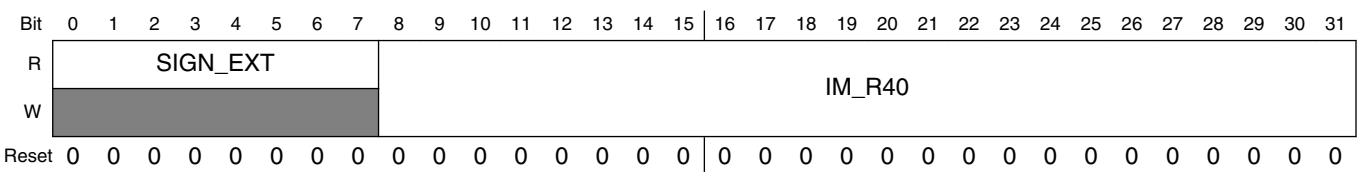


SPT_WR_R40_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R40	Real part of Work Register R40

45.8.176 Work Register R40 Imaginary (SPT_WR_R40_IM)

Address: 0h base + 394h offset = 394h

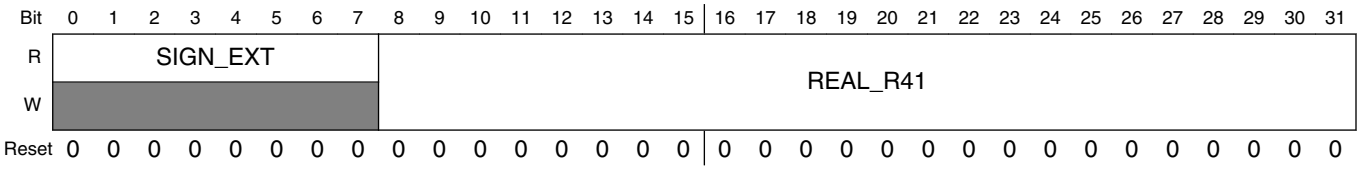


SPT_WR_R40_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R40	Imaginary part of Work Register R40

45.8.177 Work Register R41 Real (SPT_WR_R41_RE)

Address: 0h base + 398h offset = 398h

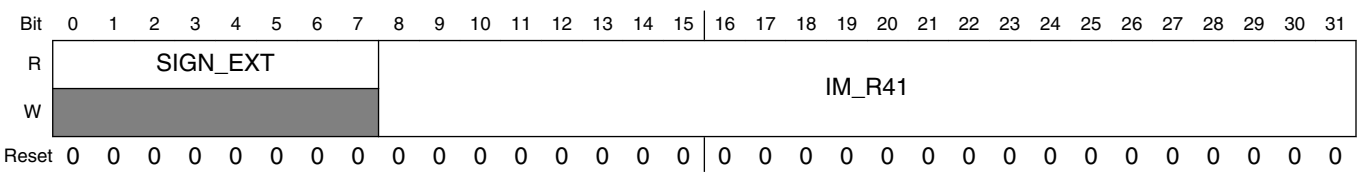


SPT_WR_R41_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R41	Real part of Work Register R41

45.8.178 Work Register R41 Imaginary (SPT_WR_R41_IM)

Address: 0h base + 39Ch offset = 39Ch

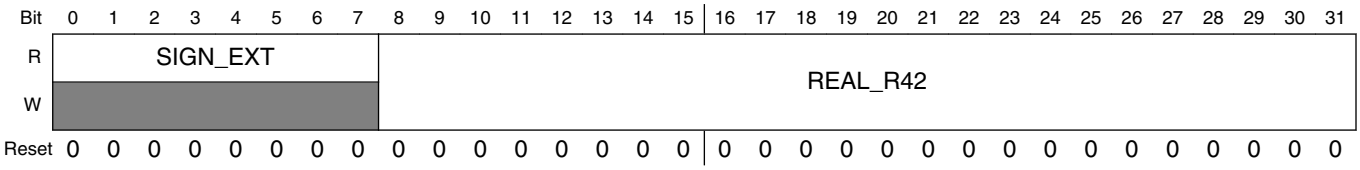


SPT_WR_R41_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R41	Imaginary part of Work Register R41

45.8.179 Work Register R42 Real (SPT_WR_R42_RE)

Address: 0h base + 3A0h offset = 3A0h

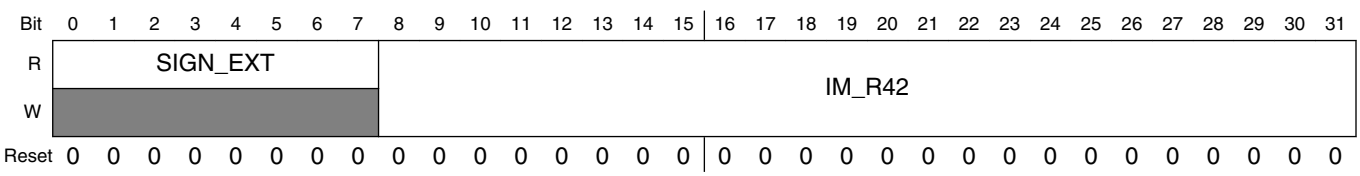


SPT_WR_R42_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R42	Real part of Work Register R42

45.8.180 Work Register R42 Imaginary (SPT_WR_R42_IM)

Address: 0h base + 3A4h offset = 3A4h

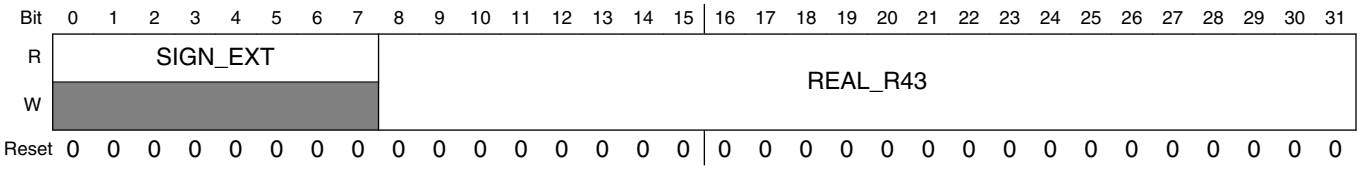


SPT_WR_R42_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R42	Imaginary part of Work Register R42

45.8.181 Work Register R43 Real (SPT_WR_R43_RE)

Address: 0h base + 3A8h offset = 3A8h

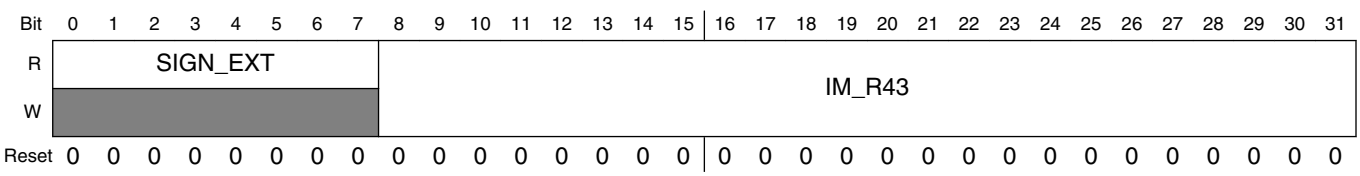


SPT_WR_R43_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R43	Real part of Work Register R43

45.8.182 Work Register R43 Imaginary (SPT_WR_R43_IM)

Address: 0h base + 3ACh offset = 3ACh

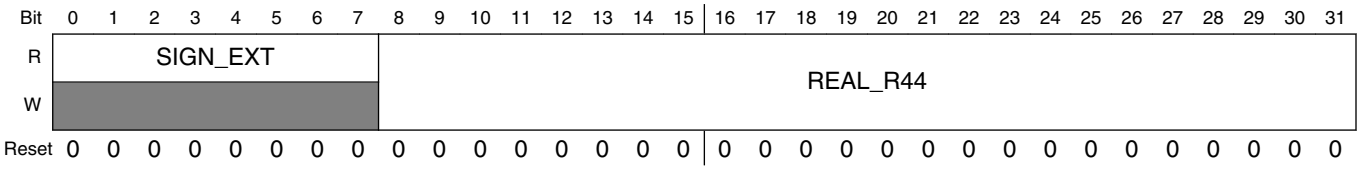


SPT_WR_R43_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R43	Imaginary part of Work Register R43

45.8.183 Work Register R44 Real (SPT_WR_R44_RE)

Address: 0h base + 3B0h offset = 3B0h

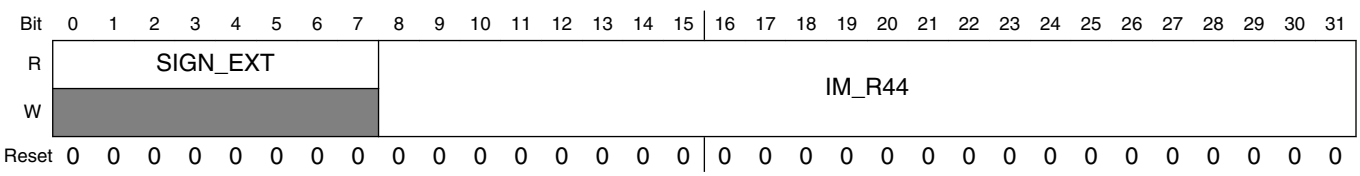


SPT_WR_R44_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R44	Real part of Work Register R44

45.8.184 Work Register R44 Imaginary (SPT_WR_R44_IM)

Address: 0h base + 3B4h offset = 3B4h

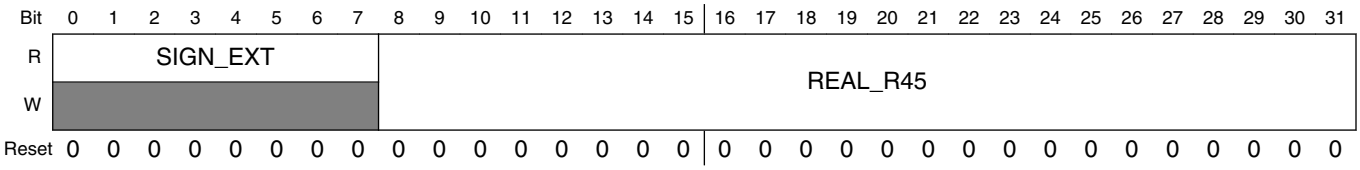


SPT_WR_R44_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R44	Imaginary part of Work Register R44

45.8.185 Work Register R45 Real (SPT_WR_R45_RE)

Address: 0h base + 3B8h offset = 3B8h

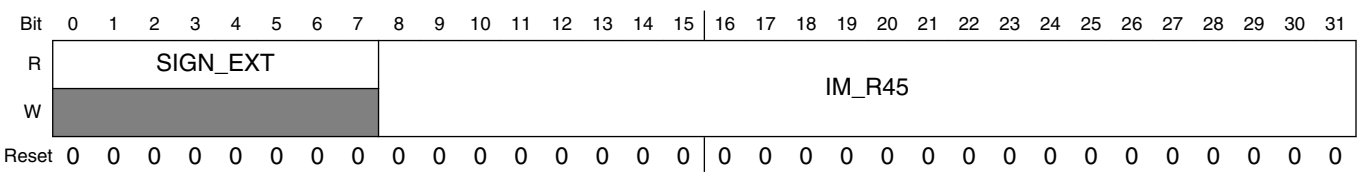


SPT_WR_R45_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R45	Real part of Work Register R45

45.8.186 Work Register R45 Imaginary (SPT_WR_R45_IM)

Address: 0h base + 3BCh offset = 3BCh

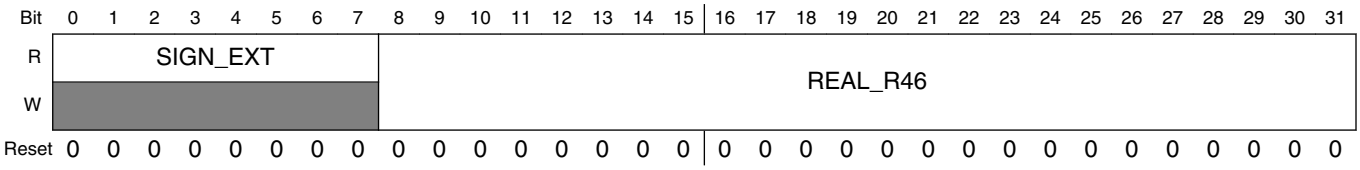


SPT_WR_R45_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R45	Imaginary part of Work Register R45

45.8.187 Work Register R46 Real (SPT_WR_R46_RE)

Address: 0h base + 3C0h offset = 3C0h

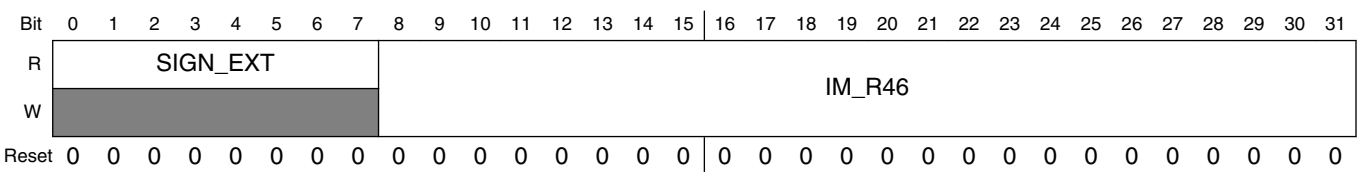


SPT_WR_R46_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R46	Real part of Work Register R46

45.8.188 Work Register R46 Imaginary (SPT_WR_R46_IM)

Address: 0h base + 3C4h offset = 3C4h

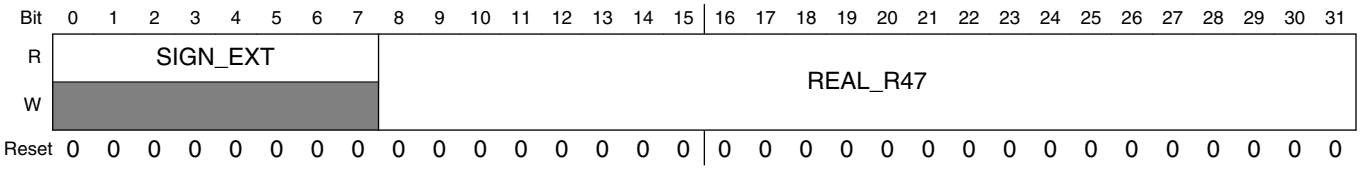


SPT_WR_R46_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R46	Imaginary part of Work Register R46

45.8.189 Work Register R47 Real (SPT_WR_R47_RE)

Address: 0h base + 3C8h offset = 3C8h

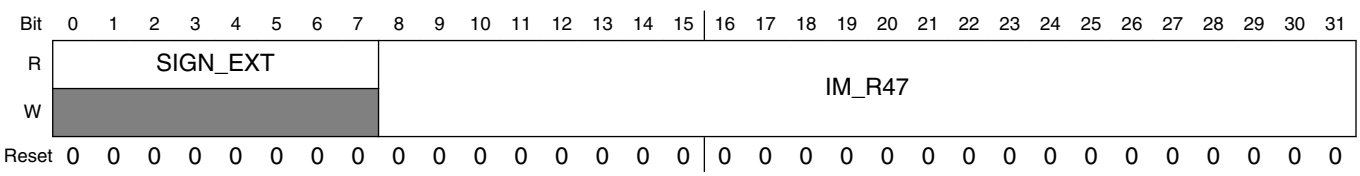


SPT_WR_R47_RE field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 REAL_R47	Real part of Work Register R47

45.8.190 Work Register R47 Imaginary (SPT_WR_R47_IM)

Address: 0h base + 3CCh offset = 3CCh



SPT_WR_R47_IM field descriptions

Field	Description
0–7 SIGN_EXT	Sign Extension This field is the Sign Extension of MSB of the work register data. Can't be written from IPS but 23rd bit which was written(from IPS or internally) will be read on these bits
8–31 IM_R47	Imaginary part of Work Register R47

45.9 Functional Description

45.9.1 Common instruction fields

The following sections define some commonly used instruction set fields. These definitions are common across all instructions using these bitfields.

45.9.1.1 Opcode

[OPCODE] - This 6-bit bitfield identifies the hardware accelerator the instruction is meant for. This bitfield occupies the same location in every instruction format. Table 45-5 lists the opcodes assigned to every hardware accelerator.

45.9.1.2 Source and Destination address

SRC_ADD/DEST_ADD - Source address [SRC_ADD] field in the instruction defines the location of the internal SPT memory resources where the input operands are to be read from. The SRC_ADD can point to Operand RAM, twiddle RAM or the work registers.

Destination address [DEST_ADD] field in the instruction defines the location of the internal SPT memory resources where the output operands are to be written to. The DEST_ADD field can point to Operand RAM, twiddle RAM or the work registers.

Refer to SPT internal memory map for details of addressing these internal memory resources.

45.9.1.3 Indirect Memory Addressing

Indirect Memory Addressing[IMA] bit provides the option of using indirect addressing for source and destination addresses. When enabled, bits 88 down to 83 of the instruction word (for instructions other than PDMA) point to a WORK REGISTER which in turn contains the source and destination addresses. Both the source and destination addresses are contained in a single WORK REGISTER - bits 39 to 24 for source address and bits 15 to 0 for destination address.

When the IMA bitfield is set, only the SRC_ADD bitfield is referred to, to get the work register address. The DEST_ADD bitfield is ignored.

In case of PDMA, Indirect Memory Addressing bit provides the option of using indirect addressing for System RAM and SPT/Operand RAM addresses. When enabled, the instruction fields[56:51] point to a WORK REGISTER which in turn contains the System RAM and SPT/Operand RAM addresses. Both the System RAM and SPT/Operand RAM addresses are contained in a single WORK REGISTER - LSBs for SPT/Operand address([23:0]) and MSBs([47:25]) for System RAM address offset.

SYS_MEM_ADR = SYS_MEM_BASE[command] + SYS_MEM_OFFS[WREG.IMAG]

45.9.1.4 Source and destination address increment

[SRC_ADD_INC]/[DEST_ADD_INC] - An option of incrementing the source and destination address is provided in most instructions. When the source address increment value is set to a value other than 0, the address for every next fetch of 8 operands is incremented by that value. Similarly, when the destination address increment value is set to a value, the address of every next write to the memory is incremented by that value.

Figure 45-4 shows how the address is incremented for subsequent fetches when the SRC_ADD_INC bitfield of the instruction is set to a value. With source or destination address increment as 0, address will not increase for subsequent fetch.

Note: The user must ensure that the source address, destination address, source address increment and destination address increment in a command word are such that the input operands are never overwritten by the output of the module.

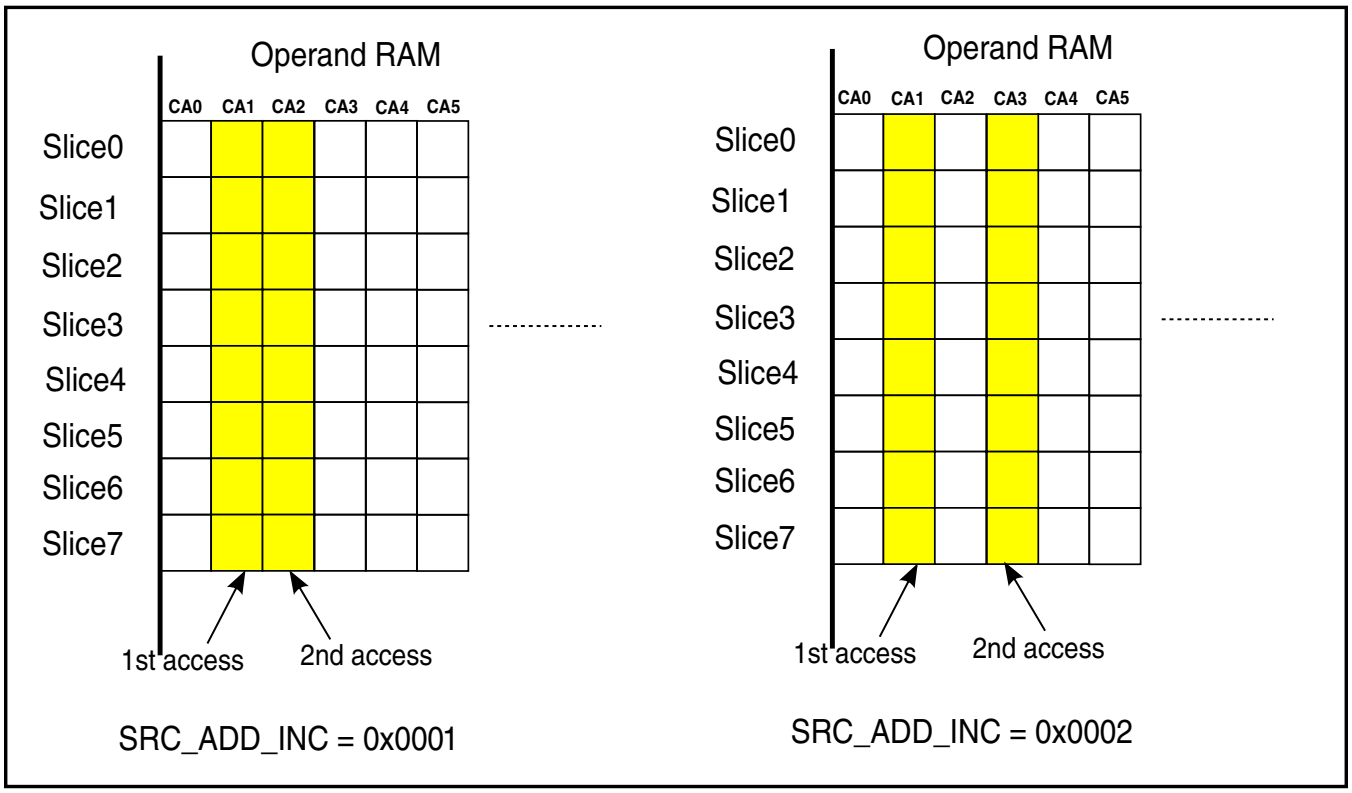


Figure 45-4. Address increment in operand RAM

45.9.1.5 Vector size

[VEC_SZ] - Vector size is defined as the number of operands on which an operation is to be performed. The size of each operand can either be one internal memory location (48bits/32bits) or half a location (24bits/16bits) in case of packed operands.

If the vector size is given as zero, then it is assumed to be 8192 operands. The definition of Vector Size is slightly different for PDMA. Refer PDMA instruction for more details.

45.9.1.6 Input Datatype/Preprocessing

The instruction bitfield IN_DATTYP is used to define the type of input data and instruction bitfield PREPROC is used to define the type of pre-processing required. The following table along with PREPROC describes the different pre-processing requirements with different types of input formats

Table 45-4. Input Datatype

Input Data	PreProcessing	Number of bits
Real	Absolute value = $\|x_k\|$	24
Complex	Sum of absolute of real and imaginary part and right shifted by 1 (| Re(x_k) | + | Im(x_k) |) >> 1	24
	Magnitude of complex operand. Refer Vector Magnitude for more details	
Log	No preprocessing	14

45.9.2 Command Sequencer

45.9.2.1 Introduction

The command sequencer reads and interprets a single instruction in the command queue. It triggers the operation specific scheduler depending on the command. A completed instruction is reported to the command queue for updating the instruction pointer. If program sequence is stalled by certain condition, the update of instruction pointer is suppressed and further program execution is suspended until this condition is cleared.

45.9.2.2 Description

The command sequencer starts operation by fetching instructions from the specified location in the system memory. The incoming instructions are stored in a command queue from where they are executed. The command queue has a size of 16 memory locations of 128-bit width. Data comes in from the CSDMA in 32 chunks of 64 bits each. It receives data from the DMA at the AHB clock frequency and gives out data to the command sequencer circuitry at the SPT clock frequency. The command sequencer does not have to wait for the command queue to be filled, it can begin execution even while the queue is being filled. An instruction start signal is given when an instruction is fired and a done signal is received when the command execution is over. Note that any illegal access to SRAM by SPT sequencer will generate PRAM crossbar fault in FCCU.

The sequencer supports early pre-fetch of instructions. If this feature is enabled, then, if the last 4 instructions in the command queue (i.e. instruction 13 to 16) are not NEXT and JUMP instructions then the sequencer requests the DMA to fetch new data at the start of the 13th instruction, else when the last instruction in the queue is issued the sequencer

also requests the DMA to refill its queue. If early pre-fetch feature is disabled then the default behavior is that when the last instruction in the queue is issued the sequencer also requests the DMA to refill its queue

Another situation where the sequencer requests a refill of the queue is when a NEXT or JUMP instruction is issued to an address outside the command queue.

It should be noted that the command sequencer queue does not have ECC protection. It also does not have parity or any other detection.

45.9.2.3 Processing states

The command sequencer finite state machine (FSM) operates in one of the following processing states:

RST: This is the state when the reset signal of the SPT is asserted.

START: When the reset is de-asserted the command sequencer enters the START state in the next clock cycle. It stays in this state until SPT_GBL_CTRL[PG_ST_CTRL] is asserted and no residual instruction fetch and no residual asynchronous PDMA instruction is in progress.

SETUP: When SPT_GBL_CTRL[PG_ST_CTRL] is asserted in the START state the sequencer moves to the SETUP state. Configuration error flags are reset to zero only in the SETUP state. Thereafter on receiving an acknowledge signal the command sequencer moves to the DEBUG state if the SPT_CS_MODE_CTRL[CS_SYS_DEBUG] is set and 'system debug' signal is asserted or if SPT_CS_MODE_CTRL[CS_IMM_DEBUG] is set, else it moves to the default RUN state.

DEBUG: This is the debug state of the SPT. There are four debugging modes in this state: halt mode, step once mode, step to next breakpoint mode and instruction jamming mode. In the halt mode the sequencer does not execute any instruction and remains halted. In the step once mode the command sequencer can step one instruction at a time; in the step to next breakpoint mode it keeps executing instructions till it reaches the next enabled breakpoint. The instruction jamming mode allows instruction execution directly through software without involving the command queue.

RUN: This is the run mode of the SPT and it is the default state of the sequencer. In this state the program in the command queue is executed till the STOP command is reached.

WAIT: The sequencer goes to the WAIT state when the WAIT command is executed, and then waits for the specified event to occur before it exits this state.

STOP: This state is reached when the STOP command is executed or when the SPT_CS_MODE_CTRL[STOP] is asserted. A 'softreset' signal is given out in this state which can be used by other modules to reset their circuitry. The sequencer remains in this state for one clock cycle and moves to the START state.

ASYNCSTOP: This state is reached when the SPT_CS_MODE_CTRL[ASYNCSTOP] signal is asserted or if an error is detected. The 'softreset' signal is also asserted in the ASYNCSTOP state. The sequencer remains in this state for one clock cycle and moves to the START state.

The command sequencer can make a state transition only on the completion of an instruction and never while an instruction is being executed, the only exception being the transition to the ASYNCSTOP state, which is in the next clock cycle. It should be noted that the SPT_CS_MODE_CTRL[STOP] or SPT_CS_MODE_CTRL[ASYNCSTOP] request has precedence over all else.

The present processing state can be assessed by SW by reading either the SPT_CS_STATUS0 configuration register or the SPT_CS_STATUS3 configuration register

Figure 45-5 shows the command sequencer FSM.

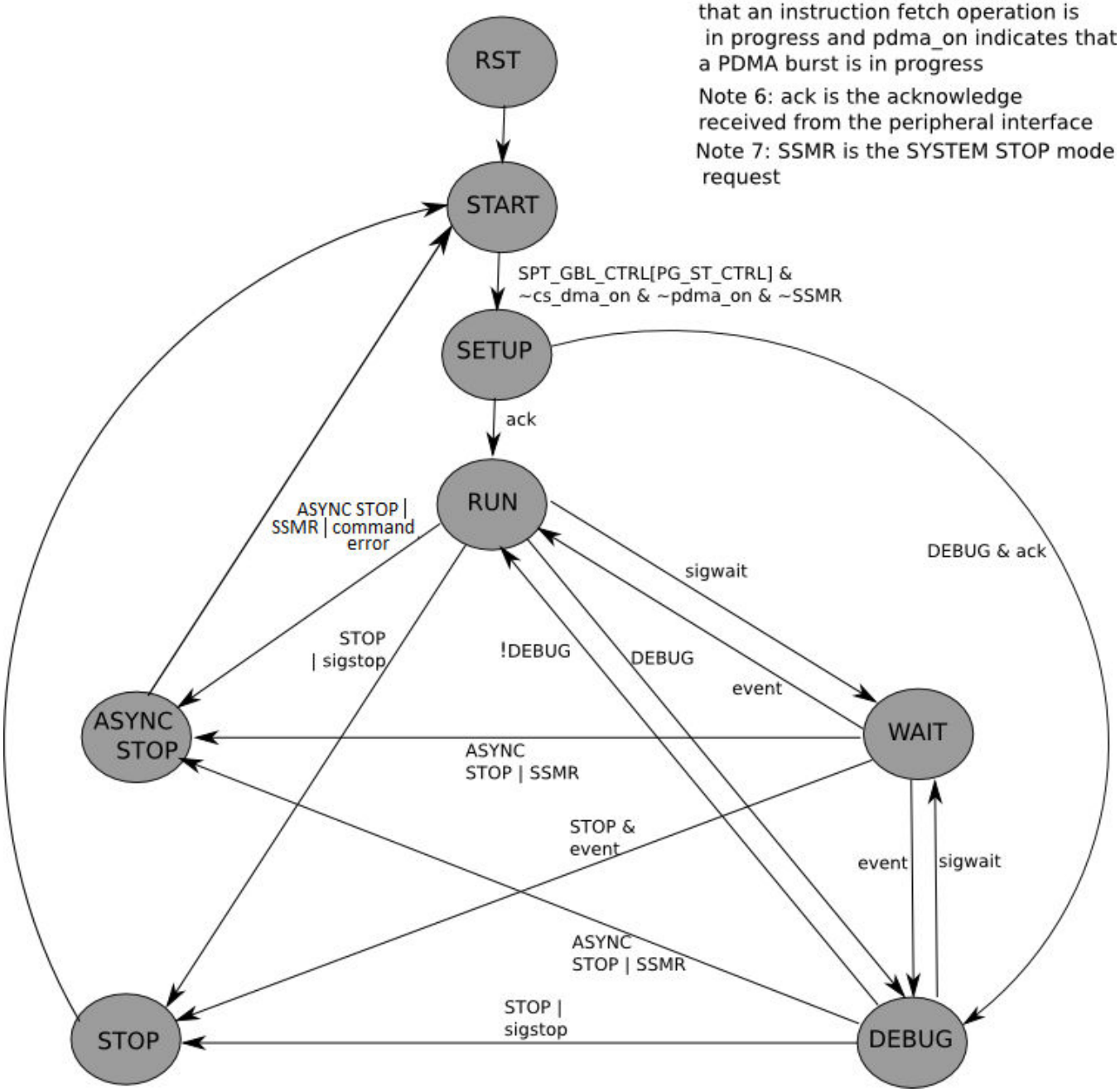


Figure 45-5. Command Sequencer FSM

45.9.2.4 Command Description

The command sequencer supports a variety of commands which are divided into two categories, operation commands and control commands.

Operation commands perform data processing operations on a given vector of data.

Control commands help to facilitate the program flow by defining loops, or checkpoints, or by performing basic data operations.

Table 45-5 provides a list of the SPT commands.

Table 45-5. List of SPT commands

Command	Opcode	Description	Details
Operation Commands			
WIN	100001	Performs a window round	Refer to FFT Instructions Format, Table 45-38, Table 45-36
RDX4	100010	Performs a Radix4 round	Refer to FFT Instructions Format, Table 45-40
RDX2	100011	Performs a Radix2 round	Refer to FFT Instructions Format, Table 45-42
HIST	100100	Performs a histogram operation	Refer to HIST instruction format
COPY	100101	Performs a copy operation	Refer to Copy Instruction Format
VMT	100110	Performs selected vector math	Refer to VMT Instruction Format
MAXS	100111	Performs maxima search operation	Refer to MAXS Instruction Format
PDMA	101000	Performs a DMA operation	Refer to PDMA instruction
FIR	101001	Performs a Finite Impulse Response (FIR) filter operation	Refer to FIR Filterand Table 45-44
SCP	101010	Scalar Product	Refer to Scalar Product
IRDX4	101011	Inverse Radix4	Refer to Inverse FFT
IRDX2	101100	Inverse Radix2	Refer to Inverse FFT
Control Commands			
SET	000001	Load target with value	Refer to SET Instruction Format
GET	000010	Copy value to work register	Refer to GET Instruction Format
ADD	000011	Add value to target	Refer to ADD Instruction Format
STOP	000100	Terminate sequence	Refer to STOP Instruction Format
LOOP	000101	Enters a loop, next commands are repeated	Refer to LOOP and NEXT Instruction Format , Table 45-13
NEXT	000110	Iterate loop	Refer to LOOP and NEXT Instruction Format , Table 45-15
SYNC	000111	Synchronize flow	Refer to SYNC Instruction Format
WAIT	001000	Suspends operation until event	Refer to WAIT Instruction Format
EVT	001001	Generates an event	Refer to EVT Instruction Format

Table continues on the next page...

Table 45-5. List of SPT commands (continued)

Command	Opcode	Description	Details
WATCHDOG	001010	Controls the watchdog timer	Refer to WATCHDOG Instruction Format
SUB	001011	Sub value from target	Refer to SUB Instruction Format
CMP	001100	Compare two values	Refer to CMP Instruction Format
JUMP	001101	Jump to forward or backward location	Refer to JUMP Instruction Format
SEL	001110	Select source to move to destination	Refer to SEL Instruction Format

The following sections provide a description of the control commands and their instruction formats.

45.9.2.4.1 SET Instruction Format

SET can be used to modify single entries in twiddle RAM, operand or work registers without involvement of the CPU. Only immediate values or work register values can be loaded. In case of the twiddle RAM being the destination the lower 16 bits from the real and imaginary part of the source operand are filled into the twiddle RAM.

If the IMA bit is set then the source address (if source is not immediate data) and destination address are taken from a work register. The SET command takes 2 cycles to execute if immediate data is the source and 3 cycles if a work register is the source. The SET command gives an error signal if the source is not immediate data or a work register, or if the source or destination is a work register with an index greater than 47. The SET command word format is shown in Table 45-6

Table 45-6. SET Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000001)						SRC	Reserved								
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved		IMA	Reserved												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMM_DAT[47:32]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table 45-6. SET Instruction Format (continued)

IMM_DAT[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMM_DAT[15:0]															

Table 45-7. SET Instruction bit description

Bits	Bit Field	Description
[121]	SRC	This 1-bit field specifies whether the source of the data to be loaded is the immediate value or a work register.
		0 – immediate data
		1 – work register
[109]	IMA	Decides whether addressing is immediate (from the instruction itself) or indirect.
		0 - Immediate addressing
		1-Indirect addressing; the address comes from a work register for both, the source and the destination. This work register index is the 6 lower bits of the SRC_ADD field.
[95:80]	SRC_ADD	This 16-bit field gives the source of the data. It is used to give the source work register when the SRC field is set.
[79:64]	DEST_ADD	This 16-bit field gives the destination address. It can be the address of the operand RAM, twiddle RAM or work register.
[47:0]	IMM_DAT	This 48-bit field gives the immediate value to be loaded.

45.9.2.4.2 GET Instruction Format

GET command can be used to copy single entries from twiddle RAM or operand RAM to work registers for access by CPU. When the source is the twiddle RAM the 32-bit data from the twiddle RAM gets stored in the lower 16 bits of the real and imaginary parts of the work register, and the upper 8 bits are sign extended. The GET command takes 3 clock cycles for execution.

If the IMA bit is set then the source address and the destination address are taken from a work register.

The GET instruction gives an error signal if the source is any other than operand RAM or twiddle RAM, or if the destination is not a work register, or if the destination is a work register with an index greater than 47. The GET instruction command word format is shown in Table 45-8

Table 45-8. GET Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000010)						Reserved									
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved		IMA	Reserved												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-9. GET Instruction bit description

Bits	Bit Field	Description
[109]	IMA	Decides whether addressing is immediate (from the instruction itself) or indirect.
		0 - Immediate addressing
		1-Indirect addressing; the address comes from a work register for both, the source and the destination. This work register index is the 6 lower bits of the SRC_ADD field.
[95:80]	SRC_ADD	This is the address of the operand RAM or twiddle RAM.
[79:64]	DEST_ADD	This is the work register address.

Figure 45-6 shows the options for SET, GET and COPY commands

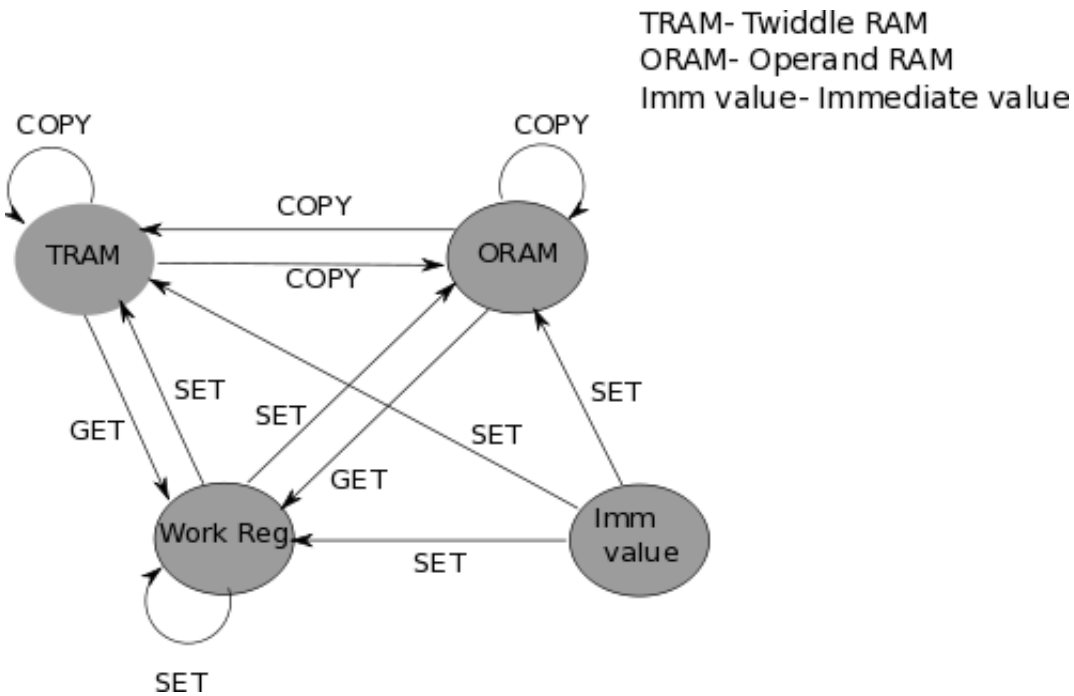


Figure 45-6. Options for SET, GET and COPY commands

45.9.2.4.3 ADD Instruction Format

ADD can be used to modify single entries in twiddle RAM, operand RAM or work registers without involvement of the CPU. Only immediate values or work register values can be added. In this operation the single entry from the operand RAM, twiddle RAM or a work register is read and the second value obtained from the immediate data or another work register is added to it and the sum(of the real as well as imaginary part) is stored back to an address in the twiddle RAM, operand RAM or a work register.

Therefore the ADD command can be summarized as

SRC1 + SRC2 -> DEST or

SRC1 + IMM_DAT -> DEST

where SRC1 and DEST can be twiddle RAM, operand RAM or a work register,

SRC2 can be a work register,

IMM_DAT is the immediate value field in the command word

The sum can be optionally shifted left by 1-bit. Also, a modulo 2^N (N = 0 to 24) operation is performed on the optionally shifted sum before it is written back to the memory.

When a 32-bit twiddle RAM entry is the source to be modified then only the lower 16 bits from the real and imaginary parts of the source are used and when the twiddle RAM is the destination only the lower 16 bits of the sums of real and imaginary parts are written back. The ADD command word format is shown in Table 45-10

The ADD instruction takes 5 cycles to complete.

An error condition is indicated if the SRC2 address is any other than of a work register.

Also error conditions are raised if for SRC1, SRC2 or DEST the address is of a work register with index greater than 47 (This takes into consideration only the 6 bits used to specify the work register index).

Table 45-10. ADD Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000011)						SRC	SHIFT	Reserved							
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved											MODULO_VAL				
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC1_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
SRC2_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMM_DAT[47:32]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMM_DAT[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMM_DAT[15:0]															

Table 45-11. ADD Instruction bit description

Bits	Bit Field	Description
[121]	SRC	This 1-bit field specifies whether the SRC2 data comes from a work register or is an immediate value.
		0 – immediate data
		1 – work register
[120]	SHIFT	Indicates whether output is left shifted by 1bit or not
		0 - Output not left shift
		1 - Output left shifted by 1-bit
[100:96]	MODULO_VAL	00000 - modulo 2^0

Table continues on the next page...

Table 45-11. ADD Instruction bit description (continued)

		00001 - modulo 2^1
		00010 - modulo 2^2
		00011 - modulo 2^3
		00100 - modulo 2^4
		00101 - modulo 2^5
		00110 - modulo 2^6
		00111 - modulo 2^7
		01000 - modulo 2^8
		01001 - modulo 2^9
		01010 - modulo 2^10
		01011 - modulo 2^11
		01100 - modulo 2^12
		01101 - modulo 2^13
		01110 - modulo 2^14
		01111 - modulo 2^15
		10000 - modulo 2^16
		10001 - modulo 2^17
		10010 - modulo 2^18
		10011 - modulo 2^19
		10100 - modulo 2^20
		10101 - modulo 2^21
		10110 - modulo 2^22
		10111 - modulo 2^23
		11000 to 11111 - modulo 2^24
[95:80]	SRC1_ADD	This field gives the SRC1 address. It can be the address of the operand RAM, twiddle RAM or work register.
[79:64]	DEST_ADD	This field gives the destination address. It can be the address of the operand RAM, twiddle RAM or work register.
[63:48]	SRC2_ADD	If the SRC field is HIGH, then this field gives the address of the work register containing SRC2 operand.
[47:0]	IMM_DAT	This 48-bit field gives the immediate value to be added.

45.9.2.4.4 STOP Instruction Format

This is the final command of a sequence of instructions. On arrival of the STOP command the sequencer goes into STOP state, and no more instructions can be executed. If an asynchronous PDMA instruction is in progress when STOP command is fired then the sequencer waits for it to complete before entering the STOP state To restart the

instruction sequence, SPT_GBL_CTRL[PG_ST_CTRL] is set, the sequencer goes to the SETUP state, then to the RUN state (or the DEBUG state) and the command execution goes to the first instruction specified by SPT_CS_PG_ST_ADDR. The command word format for STOP instruction is shown in Table 45-12

Table 45-12. STOP Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000100)						Reserved									
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

45.9.2.4.5 LOOP and NEXT Instruction Format

The LOOP command starts a loop and gives the number of iterations. There can be four levels of nested loops. If a fifth LOOP instruction is started then an error condition is indicated; an error condition also occurs if a LOOP command of zero iterations is fired. The NEXT command is used to mark the end of the loop; if a NEXT command is decoded and there is no loop in progress then also an error is raised. A new fetch is performed by the command sequencer only when the last instruction in the command queue is reached (or the 13th instruction in case of early pre-fetch) or during a NEXT instruction pointing to a location outside the span of the command queue. If the loop is contained within the command queue, then no new fetching of data into the queue is executed. LOOP and NEXT finish execution in 1 clock cycle. The command word format for LOOP instruction is shown in Table 45-13

Table 45-13. LOOP Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000101)						Reserved									

Table continues on the next page...

Table 45-13. LOOP Instruction Format (continued)

111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
LP_CNT															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-14. LOOP Instruction bit description

Bits	Bit Field	Description
[111:96]	LP_CNT	This field gives the number of iterations of the loop

The command word format for NEXT instruction is shown in Table 45-15

Table 45-15. NEXT Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000110)						Reserved									
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

45.9.2.4.6 SYNC Instruction Format

The SYNC command synchronizes the processing flow of the SPT. If independent operations were started asynchronous, e.g. a VMT and an asynchronous PDMA operation, the SYNC command enables the definition of a point in the instruction sequence where all previous scheduled commands must be finished, before the next instruction in the sequence is executed. The command word format for the SYNC instruction is shown in Table 45-16

Table 45-16. SYNC Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (000111)						Reserved									
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

45.9.2.4.7 WAIT Instruction Format

The WAIT command executes in one clock cycle and causes the sequencer to go to the WAIT state where it waits for the specified event. When the event occurs the sequencer comes back to the previous state (which could be RUN or DEBUG) and moves on to execute the subsequent commands.

Between two WAIT commands there should be a gap of >2 peripheral clock cycles. So, suppose SPT clock frequency=200 MHz and peripheral clock frequency=66.6 MHz, the ratio of SPT clock frequency to peripheral clock frequency=3. Thus the gap should be >3x2 SPT clock cycles.

There are 17 signals on which WAIT can occur. They are eight SW-generated events, six signals coming from CTE (or in case of MIPICSI2 these can be the hsync and vsync signals), chirp acquisition done, frame acquisition done and pdma transfer done. The SW-

generated events can be produced by writing to the SPT_CS_SW_EVTREG configuration register. The WAIT on these events can only be positive-edge or positive-level triggered. The WAIT on the CTE or MIPICSI2 signals can be both positive or negative, edge or level triggered.

In case of 'chirp acquisition done', 'frame acquisition done' and 'pdma transfer done' signals the WAIT can be positive edge triggered or positive level triggered. For a positive edge triggered WAIT command, the WAIT ends by directly corresponding to the awaited signal coming from the PDMA module(for pdma transfer done) or Acquisition module (for chirp acquisition done or frame acquisition done). However, if the WAIT is on a level triggered signal then the situation is a bit different.

To handle the positive level triggering case the chirp acquisition done, frame acquisition done and pdma transfer done signals they are also internally registered inside the WAIT module, and if WAIT for the event on any of these lines is on positive level triggering then it is this registered version of the event which is used to end the WAIT. But it should be noted that the internal registers are cleared when event on WAIT occurs either for positive level or even posedge event. These internal registers are also resettable through peripheral interface by writing a '1' to the SPT_EVENT_RST_CTRL configuration register. It is intended that the registered version of the event is used when there is probability of the event arriving before wait on that event starts, thereby ensuring that the event does not get missed.

Figure 45-7 shows the handling of pdma transfer done, frame acquisition done and chirp acquisition done.

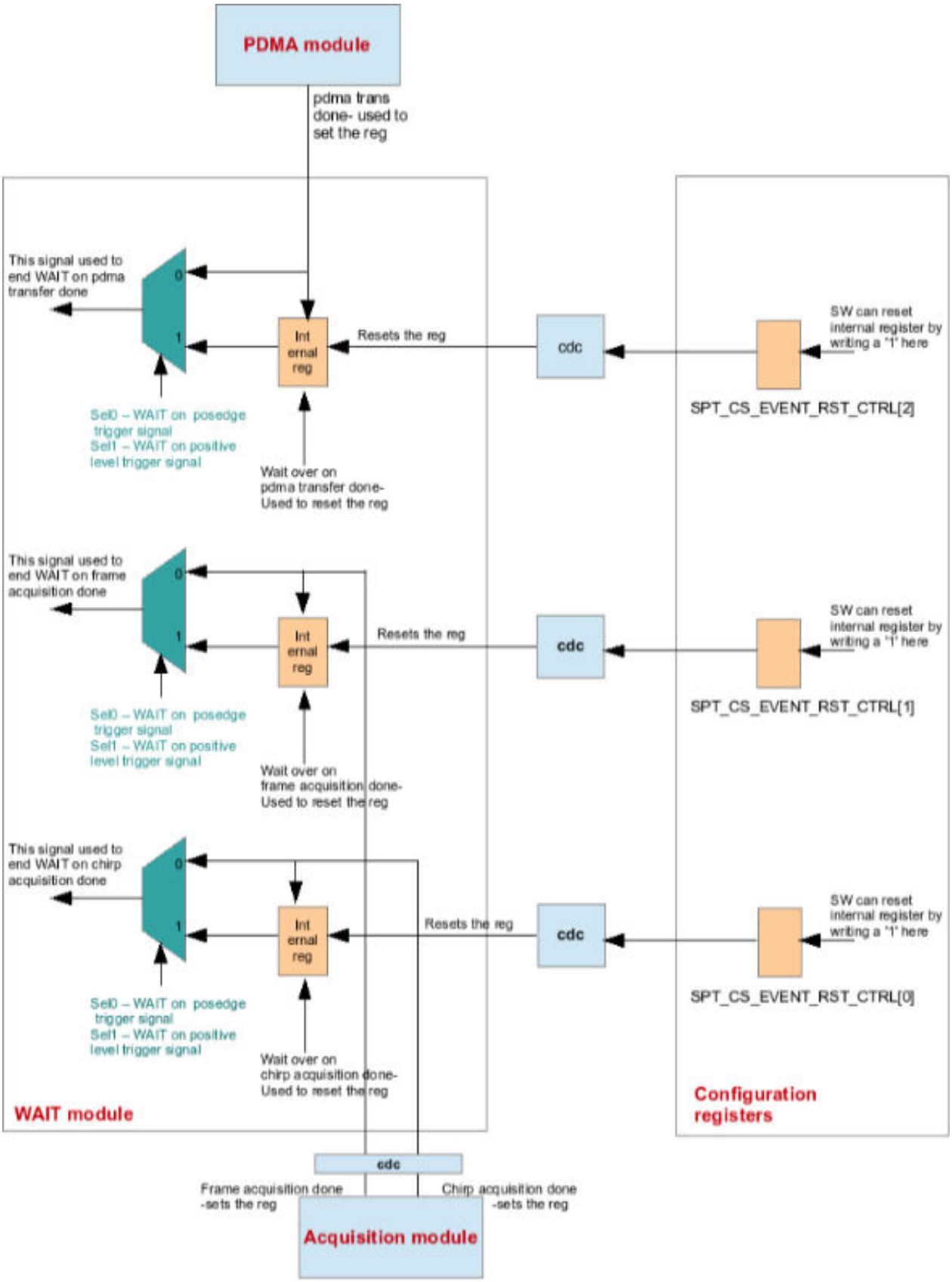


Figure 45-7. Handling of pdma transfer done, frame acquisition done and chirp acquisition done

The WAIT command word format is shown in Table 45-17

Table 45-17. WAIT Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001000)						EV_TR		Reserved			EV				
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-18. WAIT Instruction bit description:

Bits	Bit Field	Description
[121:120]	EV_TR	This 2-bit field specifies the type of event triggering
		00 – Level 0 triggered
		01 – Level 1 triggered
		10 – Positive edge triggered
		11 – Negative edge triggered
[116:112]	EV	This field specifies the awaited event
		00000-00111- Waits for SW- generated event. There are eight lines on which SW can generate events. 00000- Event awaited on LSB line 00111- Event awaited on MSB line The WAIT on these SW events can only be positive edge triggered or positive level triggered.
		01000- Event awaited on cte_evt[0]
		01001- Event awaited on cte_evt[1]
		01010- Event awaited on cte_evt[2]
		01011- Event awaited on cte_evt[3]

Table continues on the next page...

Table 45-18. WAIT Instruction bit description: (continued)

		01100- Event awaited on cte_rcs or csi_hsync
		01101- Event awaited on cte_rfs or csi_vsync
		01110- Event awaited on 'chirp acquisition done' signal. The WAIT on this event can be only positive edge triggered or positive level triggered.
		01111- Event awaited on 'frame acquisition done' signal.The WAIT on this event can be only positive edge triggered or positive level triggered.
		10000 to 11111- Event awaited on 'pdma transfer done' signal.The WAIT on this event can be only positive edge triggered or positive level triggered.

NOTE

The signals cte_evt[x] are used to signal an event from the CTE to the SPT WAIT module. These signals are also called spt_evt[x].

45.9.2.4.8 EVT Instruction Format

This commands throws up the specified event. It executes in one clock cycle. There are eight lines on which events can be thrown up. An event is issued when a '1' is written on its line and is reset when a '0' is written on it. It should be noted an event fired in the SPT domain crosses to a configuration register on the peripheral interface which is a slower domain. This enforces the constraint that there should be a gap of 12 SPT clock cycles between two events fired on the same line.

When an event is set or reset, the value is also written on the SPT_CS_EVTREG1 and SPT_CS_EVTREG2 configuration registers (these are identical configuration registers meant for two different CPU cores). So SW can read these registers to find out which of eight lines has thrown up an event. SW can also reset these registers by writing a '1' on them

The EVT command word format is shown in Table 45-19

Table 45-19. EVT Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001001)						Reserved		EV_LVL	Reserved				EV		
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96

Table continues on the next page...

Table 45-19. EVT Instruction Format (continued)

Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-20. EVT Instruction bit description

Bits	Bit Field	Description
[119]	EV_LVL	This specifies whether the event thrown out is at logic LOW or logic HIGH level
		0 - Logic LOW
		1 - Logic HIGH
[114:112]	EV	This field is the encoded event list. 000- event thrown up on LSB line 111- event thrown up on MSB line

Figure 45-8 shows the synchronization of the CPU with SPT using the WAIT and EVT commands.

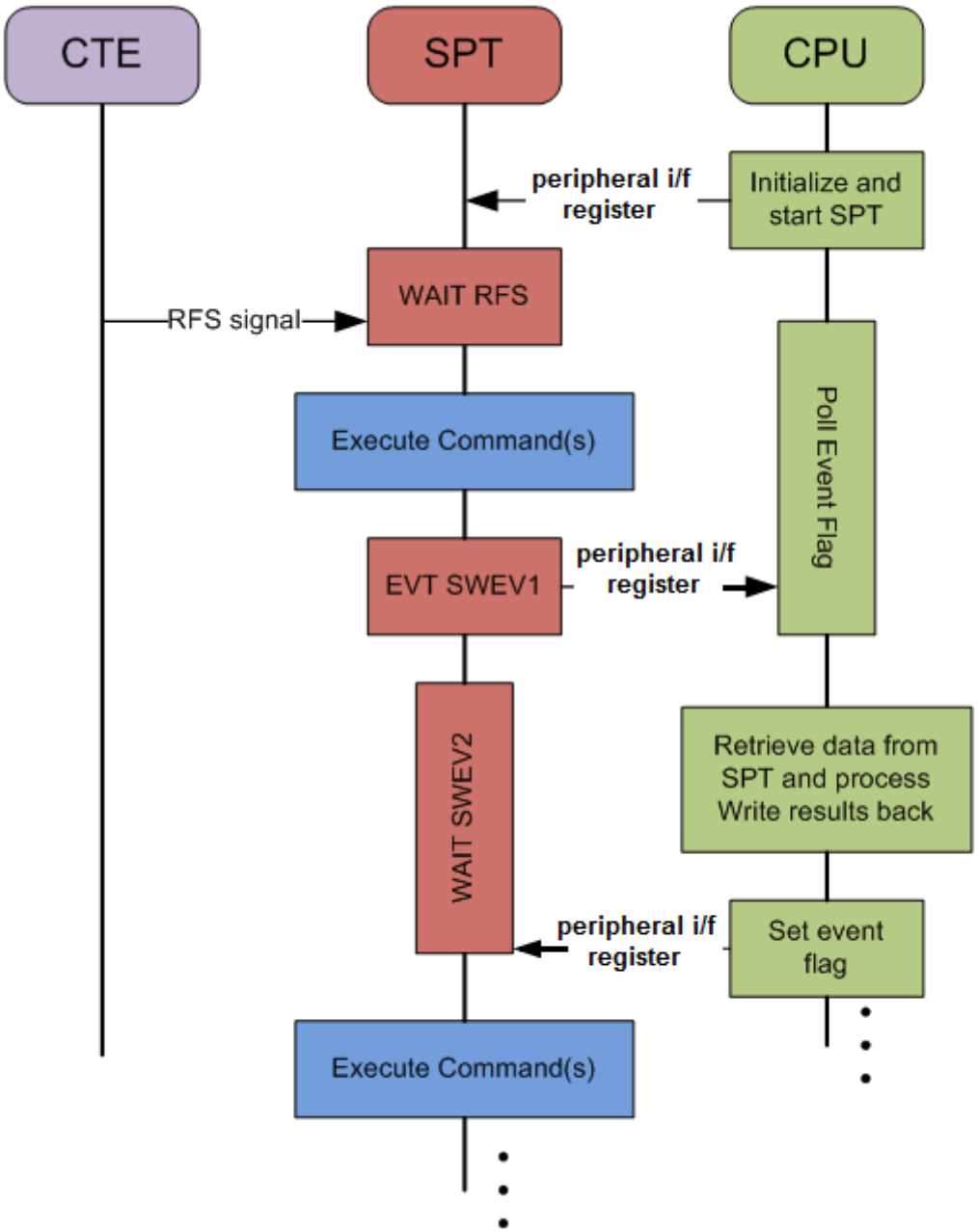


Figure 45-8. CPU-SPT synchronization using WAIT and EVT commands.

45.9.2.4.9 WATCHDOG Instruction Format

The watchdog requires 24 bits. It can be loaded and started to count down, or it can be halted (stopped) or it can be reset with the WATCHDOG command. This is the count mode, and in this mode when the counter counts down to zero an interrupt can be generated and a bit in the SPT_CS_STATUS0 configuration register is set.

The watchdog counter value is reflected in the configuration map in the SPT_CS_WD_COUNT register, to be read by SW. Due to this clock domain crossing from the SPT domain to the slower configuration map domain it is required that the programmed initial value for the watchdog timer be 32 or greater.

Additionally, the watchdog can work in the event mode where it can be loaded with a value and used as a flag. In this case a CTE event (or in case of MIPICSI2 the vsync and hsync events) is specified and when the event arrives the watchdog is reset to zero. SW can enquire about the awaited event by reading the SPT_CS_STATUS2 configuration register.

NOTE

When the watchdog circuit is waiting for an event, it is not recommended to fire another 'event mode' watchdog command until the wait for the first event is over.

The WATCHDOG instruction executes in 1 clock cycle. It should be noted that the watchdog timer is immediately reset upon getting the 'soft reset' signal from the command sequencer. The command word format for WATCHDOG instruction is shown in Table 45-21

Table 45-21. WATCHDOG Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001010)						EV_TR		Reserved		OP		Reserved	EV		
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CNT_IN_VAL[23:16]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT_IN_VAL[15:0]															

Table 45-22. WATCHDOG Instruction bit description

Bits	Bit Field	Description

Table continues on the next page...

Table 45-22. WATCHDOG Instruction bit description (continued)

[121:120]	EV_TR	This 2-bit field gives the type of event triggering.
		00 - Level 0 triggered
		01 - Level 1 triggered
		10 - Positive edge triggered
		11 - Negative edge triggered
[117:116]	OP	Decides the watchdog operation and mode
		00- Watchdog counter is loaded with the initialization value, and the counter starts counting down. This is the count mode
		01- Stops the counter in the count mode
		10- Resets the counter in the count mode
		11- The watchdog counter is enabled to be used as a flag. The counter resets when the specified event occurs. This is the event mode
[114:112]	EV	This field specifies the event which resets the timer
		000- cte_evt[0]
		001- cte_evt[1]
		010- cte_evt[2]
		011- cte_evt[3]
		100- cte_rcs or csi_hsync
		101 to 111- cte_rfs or csi_vsync
[23:0]	CNT_IN_VAL	This 24-bit field gives the value to which the counter is initialized.

45.9.2.4.10 SUB Instruction Format

SUB can be used to modify single entries in twiddle RAM, operand RAM or work registers without involvement of the CPU. Only immediate values or work register values can be subtracted. In this operation the single entry from the operand RAM, twiddle RAM or a work register is read and the second value obtained from the immediate data or another work register is subtracted from it and the difference (of the real as well as imaginary part) is stored back to an address in the twiddle RAM, operand RAM or a work register.

Therefore the SUB command can be summarized as

SRC1 - SRC2 -> DEST or

SRC1 - IMM_DAT -> DEST

where SRC1 and DEST can be twiddle RAM, operand RAM or a work register,

SRC2 can be a work register,

IMM_DAT is the immediate value field in the command word

The difference can be optionally shifted left by 1-bit. Also, a modulo 2^N (N = 0 to 24) operation is performed on the optionally shifted difference before it is written back to the memory.

When a 32-bit twiddle RAM entry is the source to be modified then only the lower 16 bits from the real and imaginary parts of the source are used and when the twiddle RAM is the destination only the lower 16 bits of the differences of real and imaginary parts are written back. The SUB command word format is shown in Table 45-23

The SUB instruction takes 5 cycles to complete.

An error condition is indicated if the SRC2 address is any other than of a work register.

Also error conditions are raised if for SRC1, SRC2 or DEST the address is of a work register with index greater than 47 (This takes into consideration only the 6 bits used to specify the work register index).

Table 45-23. SUB Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001011)						SRC	SHIFT	Reserved							
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved											MODULO_VAL				
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC1_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
SRC2_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMM_DAT[47:32]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMM_DAT[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMM_DAT[15:0]															

Table 45-24. SUB Instruction bit description

Bits	Bit Field	Description

Table continues on the next page...

Table 45-24. SUB Instruction bit description (continued)

[121]	SRC	This 1-bit field specifies whether the SRC2 data comes from a work register or is an immediate value.
		0 – immediate data
		1 – work register
[120]	SHIFT	Indicates whether output is left shifted by 1bit or not
		0 - Output not left shift
		1 - Output left shifted by 1-bit
[100:96]	MODULO_VAL	00000 - modulo 2^0
		00001 - modulo 2^1
		00010 - modulo 2^2
		00011 - modulo 2^3
		00100 - modulo 2^4
		00101 - modulo 2^5
		00110 - modulo 2^6
		00111 - modulo 2^7
		01000 - modulo 2^8
		01001 - modulo 2^9
		01010 - modulo 2^10
		01011 - modulo 2^11
		01100 - modulo 2^12
		01101 - modulo 2^13
		01110 - modulo 2^14
		01111 - modulo 2^15
		10000 - modulo 2^16
		10001 - modulo 2^17
		10010 - modulo 2^18
		10011 - modulo 2^19
		10100 - modulo 2^20
		10101 - modulo 2^21
		10110 - modulo 2^22
		10111 - modulo 2^23
		11000 to 11111 - modulo 2^24
[95:80]	SRC1_ADD	This field gives the SRC1 address. It can be the address of the operand RAM, twiddle RAM or work register.
[79:64]	DEST_ADD	This field gives the destination address. It can be the address of the operand RAM, twiddle RAM or work register.
[63:48]	SRC2_ADD	If the SRC field is HIGH, then this field gives the address of the work register containing SRC2 operand.

Table continues on the next page...

Table 45-24. SUB Instruction bit description (continued)

[47:0]	IMM_DAT	This 48-bit field gives the immediate value to be subtracted.

45.9.2.4.11 CMP Instruction Format

CMP can be used to enhance the decision making ability of SPT without involvement of the CPU. In this operation the first value is the single entry read from the operand RAM, twiddle RAM or a work register and the second value is obtained from the immediate data or another work register. This second value is subtracted from it and the result is stored back to the destination address in the twiddle RAM, operand RAM or a work register.

This result consists of status bits which indicate whether the first value is i) greater than, ii) greater than or equal to, iii) less than, iv) less than or equal to, v) or equal to the second value. This result is stored for real as well as imaginary part. It is intended that the CMP instruction will be used together with the SEL or JUMP instruction, where the result produced by CMP will be utilized by SEL or JUMP instruction.

Therefore the CMP command can be summarized as

SRC1 - SRC2 -> DEST or

SRC1 - IMM_DAT -> DEST

where SRC1 and DEST can be twiddle RAM, operand RAM or a work register,

SRC2 can be a work register,

IMM_DAT is the immediate value field in the command word

It should be noted that both the comparisons are considered to be a signed comparison.

The resulting status flags are stored in the destination in the following format:

Table 45-25. Format of result stored in the destination

Bit position in destination	Description of corresponding status bit
0	Real part of SRC1 == Real part of SRC2
1	Real part of SRC1 <= Real part of SRC2
2	Real part of SRC1 < Real part of SRC2
3	Real part of SRC1 >= Real part of SRC2
4	Real part of SRC1 > Real part of SRC2
5	Imaginary part of SRC1 == Imaginary part of SRC2
6	Imaginary part of SRC1 <= Imaginary part of SRC2

Table continues on the next page...

Table 45-25. Format of result stored in the destination (continued)

7	Imaginary part of SRC1 < Imaginary part of SRC2
8	Imaginary part of SRC1 >= Imaginary part of SRC2
9	Imaginary part of SRC1 > Imaginary part of SRC2
Others	0

When a 32-bit twiddle RAM entry is the source then only the lower 16 bits from the real and imaginary parts of the source are used for comparison. The CMP command word format is shown in Table 45-26

The CMP instruction takes 5 cycles to complete.

An error condition is indicated if the SRC2 address is any other than of a work register and the SRC field is HIGH.

Also error conditions are raised if for SRC1, SRC2 or DEST the address is of a work register with index greater than 47 (This takes into consideration only the 6 bits used to specify the work register index).

Table 45-26. CMP Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001100)						SRC	Reserved								
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC1_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
SRC2_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMM_DAT[47:32]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMM_DAT[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMM_DAT[15:0]															

Table 45-27. CMP Instruction bit description

Bits	Bit Field	Description
[121]	SRC	This 1-bit field specifies whether the SRC2 data comes from a work register or is an immediate value.

Table continues on the next page...

Table 45-27. CMP Instruction bit description (continued)

		0 – immediate data
		1 – work register
[95:80]	SRC1_ADD	This field gives the SRC1 address. It can be the address of the operand RAM, twiddle RAM or work register.
[79:64]	DEST_ADD	This field gives the destination address. It can be the address of the operand RAM, twiddle RAM or work register.
[63:48]	SRC2_ADD	If the SRC field is HIGH, then this field gives the address of the work register containing SRC2 operand.
[47:0]	IMM_DAT	This 48-bit field gives the immediate value to be added.

45.9.2.4.12 JUMP Instruction Format

The JUMP command can be used for branching or to form looping structures in the code. Unconditional (forced) as well as conditional jump operations are supported. In a conditional JUMP the specified bit of the specified WR is read and if the bit is SET then a jump operation is performed to the jump address, else the program moves to the next command.

It is the responsibility of the user to ensure that the instruction format is correct. It is also the responsibility of the user to ensure that the destination provided for the JUMP command does not violate the program structure e.g. infinite loops, or JUMP to a location outside the program, or JUMP into or outside of a loop.

This instruction is intended to be used in conjunction with the CMP command. The result of the CMP command can be stored in a WR, and the JUMP instruction reads this value stored in the specified WR .

The status flags from the previous CMP instruction are checked by JUMP command according to the following format:

Table 45-28. Format of status flags checked by JUMP instruction

Bit position in WR	Description of corresponding status bit
0	Real part of previous CMP instruction shows that result of the instruction is 'equal to'.
1	Real part of previous CMP instruction shows that result of the instruction is 'less than or equal to'.
2	Real part of previous CMP instruction shows that result of the instruction is 'less than'.
3	Real part of previous CMP instruction shows that result of the instruction is 'greater than or equal to'.

Table continues on the next page...

Table 45-28. Format of status flags checked by JUMP instruction (continued)

4	Real part of previous CMP instruction shows that result of the instruction is 'greater than'.
5	Imaginary part of previous CMP instruction shows that result of the instruction is 'equal to'.
6	Imaginary part of previous CMP instruction shows that result of the instruction is 'less than or equal to'.
7	Imaginary part of previous CMP instruction shows that result of the instruction is 'less than'.
8	Imaginary part of previous CMP instruction shows that result of the instruction is 'greater than or equal to'.
9-47	Imaginary part of previous CMP instruction shows that result of the instruction is 'greater than'.

The JUMP command word format is shown in Table 45-29

The JUMP instruction takes 2 cycles to complete.

Table 45-29. JUMP Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001101)						COND_TYPE	COND_SEL				Reserved				
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved							SEL_WR						Reserved		
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
Reserved															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
JUMP_ADDR[31:16]															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
JUMP_ADDR[15:0]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-30. JUMP Instruction bit description

Bits	Bit Field	Description
[121]	COND_TYPE	Decides whether the jump operation is a conditional or unconditional jump
		0 – Unconditional jump

Table continues on the next page...

Table 45-30. JUMP Instruction bit description (continued)

		1 – Conditional jump
[120:117]	COND_SEL	This bit specifies the WR bit to be read
		0- Bit 0 is to be read
		1- Bit 1 is to be read
		2- Bit 2 is to be read
		3- Bit 3 is to be read
		4- Bit 4 is to be read
		5- Bit 5 is to be read
		6- Bit 6 is to be read
		7- Bit 7 is to be read
		8- Bit 8 is to be read
		9 to 15- Bit 9 is to be read
[88:83]	SEL_WR	This field selects the WR number to be read. The value of this field can be from 0 to 47.
[63:32]	JUMP_ADDR	This is the jump offset location. It is a 32-bit relative memory location, relative to the memory location at which the JUMP command is stored. This offset is stored in a 2's complement format.

45.9.2.4.13 SEL Instruction Format

The SEL command can read one of the three specified source addresses and store it in the destination. Which of the three sources is to be read is dependent on the result of the previous CMP instruction, and hence this instruction is intended to be used in conjunction with the CMP command. The result of the CMP command can be stored in a WR, and the SEL instruction reads this value stored in the specified WR (either the real part or the imaginary part). If the result stored in the WR specifies that the previous result was 'greater than' then SRC3 is read, if the previous result was 'equal to' then SRC2 is read, else SRC1 is read and then written to the destination address.

Therefore the SEL command can be summarized as

if comparison result == 'greater than' then SRC3 -> DEST

else if comparison result == 'equal to' SRC2 -> DEST

else SRC1 -> DEST

where SRC1, SRC2, SRC3 and DEST can be twiddle RAM, operand RAM or a work register,

The status flags from the previous CMP instruction are checked by SEL command according to the following format:

Table 45-31. Format of status flags checked by SEL instruction

Bit position in WR	Description of corresponding status bit
0	Real part of previous CMP instruction shows that result of the CMP instruction is 'equal to'. Hence SRC2 is transferred to destination.
2	Real part of previous CMP instruction shows that result of the instruction is 'less than'. Hence SRC1 is transferred to destination.
4	Real part of previous CMP instruction shows that result of the instruction is 'greater than'. Hence SRC3 is transferred to destination.
5	Imaginary part of previous CMP instruction shows that result of the instruction is 'equal to'. Hence SRC2 is transferred to destination.
7	Imaginary part of previous CMP instruction shows that result of the instruction is 'less than'. Hence SRC1 is transferred to destination.
9	Imaginary part of previous CMP instruction shows that result of the instruction is 'greater than'. Hence SRC3 is transferred to destination.
Others	Not used by SEL command

When a 32-bit twiddle RAM entry is the selected source to be transferred to destination then only the lower 16 bits from the real and imaginary parts of the source are used and when the twiddle RAM is the destination only the lower 16 bits of the differences of real and imaginary parts are written back. The SEL command word format is shown in Table 45-32

The SEL instruction takes 5 cycles to complete.

An error conditions are raised if for SRC1, SRC2, SRC3 or DEST the address is of a work register with index greater than 47 (This takes into consideration only the 6 bits used to specify the work register index).

An error is also specified if the SEL_WR field which selects the WR to be read is greater than 47.

Table 45-32. SEL Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE (001110)						RE_IM_B	Reserved								
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved															

Table continues on the next page...

Table 45-32. SEL Instruction Format (continued)

95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
Reserved							SEL_WR						Reserved		
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
SRC2_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC3_ADDR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRC1_ADDR															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Table 45-33. SEL Instruction bit description

Bits	Bit Field	Description
[121]	RE_IM_B	Decides whether the real or imaginary part is selected for deciding the source to be transferred to destination
		0 – Imaginary part is to be used for selection
		1 – Real part is to be used for selection
[88:83]	SEL_WR	This field gives the index of the WR to be read
[79:64]	DEST_ADD	This field gives the destination address. It can be the address of the operand RAM, twiddle RAM or work register. In case the destination is a WR then bits 72 down to 67 specify the index of WR.
[63:48]	SRC2_ADD	This field gives the SRC2 address. It can be the address of the operand RAM, twiddle RAM or work register. In case SRC2 is a WR then bits 56 down to 51 specify the index of WR.
[47:32]	SRC3_ADD	This field gives the SRC3 address. It can be the address of the operand RAM, twiddle RAM or work register. In case SRC3 is a WR then bits 40 down to 35 specify the index of WR.
[31:16]	SRC1_ADD	This field gives the SRC1 address. It can be the address of the operand RAM, twiddle RAM or work register. In case SRC1 is a WR then bits 24 down to 19 specify the index of WR.

45.9.2.5 Command Sequencer and PDMA instruction

The command sequencer executes the commands in a sequential manner i.e. only after the present command is executed is the subsequent command fired. The exception here is the PDMA command. There are two types of PDMA commands, synchronous PDMA command and asynchronous PDMA command. A synchronous PDMA command, like any other instruction, is strictly sequential. However the asynchronous PDMA instruction may be executed concurrently along with any other instruction.

Case 1: An instruction (other than PDMA) is being executed and the subsequent instruction is an asynchronous PDMA instruction. In this situation the instruction will finish execution before the PDMA instruction is started.

Case 2: An asynchronous PDMA instruction is being executed and the subsequent instruction is any instruction other than PDMA. Here, the command sequencer does not wait for the PDMA instruction to complete and fires the following instruction.

Case 3: An asynchronous PDMA instruction followed by another PDMA instruction. In this case the first asynchronous PDMA instruction will finish execution before the second PDMA instruction is executed.

45.9.2.6 Error Handling

Errors are divided into two categories, command errors and non-command errors. The command errors are those which are caused by wrong command word options or even wrong order of instructions (like a NEXT instruction being decoded before a loop has begun). All the other error conditions (like parity error, etc.), exceptions or overflows are grouped under non-command errors.

Whenever a command error occurs the command sequencer goes to the ASYNCSTOP state and an error bit is set in the configuration map. From there it goes to the START state. To restart the execution of the program the CPU then has to assert the SPT_GBL_CTRL[PG_ST_CTRL]. On such a restart the sequencer goes to the SETUP state and the error bit is reset. Whenever a non-command error occurs, the sequencer does not go to the ASYNCSTOP state but just sets a bit in the configuration map and the command sequencer moves on.

Another important register is the interrupt enable register. When a bit of this register is set, it means that the error bit corresponding to it causes an interrupt; if the interrupt enable register bit is reset then there is no interrupt to the CPU. In such a condition the CPU would have to poll the error bits periodically to catch an error.

In case of a command error SPT_CS_ERR_INST_ADDR is filled with the address of the instruction causing the error and the corresponding error configuration bit is raised. All error bits are reset in the SETUP state. All error bits are resettable by SW.

The CPU can stop the program (in the RUN or DEBUG state) flow by setting SPT_CS_MODE_CTRL[STOP]. Here the sequencer goes to the STOP state after the current instruction has finished executing. The CPU can also cause an asynchronous STOP by setting the SPT_CS_MODE_CTRL[ASYNCSTOP] and the sequencer goes to the ASYNCSTOP state; this is an asynchronous soft reset and so does not wait for the current instruction to complete.

The command errors correspond to the following configuration register bits:

- SPT_CS_STATUS1[ILL_OPCODE]
- SPT_CS_STATUS1[ILL_LOOP]
- SPT_CS_STATUS1[ILL_0CNTLOOP]
- SPT_CS_STATUS1[ILL_NEXT]
- SPT_CS_STATUS1[ILL_SET]
- SPT_CS_STATUS1[ILL_GET]
- SPT_CS_STATUS1[ILL_ADD]
- SPT_CS_STATUS1[JAM_ILL_OPCODE]
- SPT_CS_STATUS1[JAM_ILL_SYNC]
- SPT_CS_STATUS1[JAM_ILL_LOOP]
- SPT_CS_STATUS1[JAM_ILL_NEXT]
- SPT_CS_STATUS1[JAM_ILL_JUMP]
- SPT_DMA_ERR_STATUS[ILLEGAL_COMMAND_ERR] (this error will also show up as SPT_CS_STATUS1[ILL_OPCODE])
- SPT_DMA_ERR_STATUS[DMA_COMMAND_ERR]
- SPT_HW_ACC_ERR_STATUS[VMT_TWI_DVS_ERR]
- SPT_HW_ACC_ERR_STATUS[VMT_SS_CB_ERR]
- SPT_HW_ACC_ERR_STATUS[VMT_CON_RL_ERR]
- SPT_HW_ACC_ERR_STATUS[HIST_INVALID_WR_ACCESS_ERR]
- SPT_HW_ACC_ERR_STATUS[HIST_VECLEN_NOT_MULTOF8_ERR]
- SPT_HW_ACC_ERR_STATUS[HIST_SRC_NOT_OR_ERR]
- SPT_HW_ACC_ERR_STATUS[HIST_SRCADDR_NOT_MULTOF8_ERR]
- SPT_HW_ACC_ERR_STATUS[HIST_DESTADDR_NOT_MULTOF8_ERR]
- SPT_HW_ACC_ERR_STATUS[COPY_IP_CMD_ERR]
- SPT_HW_ACC_ERR_STATUS[MAXS_IP_CMD_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_ILL_SHFTVAL]
- SPT_HW_ACC_ERR_STATUS[FFT_WIN_RND_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_RDX4_RND_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_QE_VL_OVS_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_TW_OVS_ERR]

- SPT_HW_ACC_ERR_STATUS[FFT_MULT_COEFF_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_MULT_COEFF2_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_OPR_ADDR_ERR]
- SPT_HW_ACC_ERR_STATUS[FFT_RDX2_RND_ERR]

The non-command errors (and exceptions and overflows) correspond to the following register bits:

- SPT_MEM_ERR_STATUS[WR_LCK_VIOL]
- SPT_MEM_ERR_STATUS[TR_PAR_ERR]
- SPT_MEM_ERR_STATUS[WR_OR_3_ACC_CONTN]
- SPT_MEM_ERR_STATUS[WR_OR_2_ACC_CONTN]
- SPT_MEM_ERR_STATUS[OR_DMA_PAR_ERR]
- SPT_MEM_ERR_STATUS[OR_MOD_PAR_ERR]
- SPT_DMA_ERR_STATUS[SDMA_WR_ERR]
- SPT_DMA_ERR_STATUS[SDMA_RD_ERR]
- SPT_DMA_ERR_STATUS[PDMA_AHB_ERR]
- SPT_DMA_ERR_STATUS[SDMA_AHB_ERR]
- SPT_DMA_ERR_STATUS[CS_AHB_ERR]
- SPT_DMA_ERR_STATUS[SDMA_RD_ERR_BYP]
- SPT_DMA_ERR_STATUS[SDMA_WR_ERR_BYP]
- SPT_DMA_ERR_STATUS[LINE_LEN_EXCD_ERR]
- SPT_DMA_ERR_STATUS[LINE_LEN_EXCD_BYP_ERR]
- SPT_DMA_ERR_STATUS[INCMPLT_SMPL_RCVD]
- SPT_HW_ACC_ERR_STATUS[VMT_SHIFT_OVF_ERR]
- All bits of SPT_HIST_OVF_STATUS0
- All bits of SPT_HIST_OVF_STATUS1

NOTE

During reading of OPRAM/TRAM, the reads happen on all slices unconditionally. Therefore even if the data of a particular memory location is not used but the memory location has a parity error then the parity error will be flagged. In case of FFT commands like RDX4 and RDX2 due to design optimization eight locations before the start of the twiddle address will also be read if quadrature extension is enabled. If there is a valid parity error on any of these eight locations then the parity error will be flagged. Thus, for RDX4 and RDX2 commands it is recommended that the entire TRAM is initialized, and in general both OPRAM and TRAM are initialized before being used.

NOTE

In case of multiple parity errors occurring during memory read, the address of the first memory location encountering parity error is sent out to MEMU.

NOTE

When SPT_GBL_CTRL[SPT_EN] is LOW all the status registers must be ignored and before SPT_GBL_CTRL[SPT_EN] is deasserted all the interrupts must be disabled

45.9.2.7 Debug Strategy

45.9.2.7.1 Entering the debug state and various debugging modes

The sequencer can go to the DEBUG state directly from the SETUP state or from the RUN state. The sequencer enters the DEBUG state by the following ways:

1. The sequencer is in the RUN state and SPT_CS_MODE_CTRL[CS_IMM_DEBUG] is set. Then the sequencer moves to the DEBUG state upon completion of current command.
2. The sequencer is in the RUN state and the SPT_CS_MODE_CTRL[CS_BKPT_DEBUG] field is set in the configuration map. Then the command sequencer can move from the RUN state to the DEBUG state when it hits any of the four breakpoints. These breakpoints can be enabled or disabled through the configuration map. There is also the option to automatically disable a breakpoint after a break occurs at that particular breakpoint.
3. The sequencer is in the RUN state and the SPT_CS_MODE_CTRL[CS_SYS_DEBUG] field is set in the configuration map. The sequencer moves from the RUN state to the DEBUG state in synchronization with the CPU. When the CPU hits a breakpoint it raises the ‘system debug’ signal and the sequencer enters the debug state.
4. The sequencer can go directly from the SETUP state to the DEBUG state if the SPT_CS_MODE_CTRL[CS_SYS_DEBUG] is set and 'system debug' signal is asserted or if SPT_CS_MODE_CTRL[CS_IMM_DEBUG] is set.

When both SPT_CS_MODE_CTRL[CS_BKPT_DEBUG] and SPT_CS_MODE_CTRL[CS_SYS_DEBUG] fields are set then whether the command sequencer enters the DEBUG state due to its own breakpoints or due to the system debug signal depends on which occurs first. If SPT_CS_MODE_CTRL[CS_IMM_DEBUG] field is set then this has priority and the sequencer enters debug state when the present command is done.

When in the DEBUG state the sequencer supports the following four sub-modes as determined by SPT_CS_MODE_CTRL[DEBUG_MD].

SPT_CS_MODE_CTRL[DEBUG_MD] = 00b: The sequencer remains halted.

SPT_CS_MODE_CTRL[DEBUG_MD] = 01b: Step once mode (sequencer steps ahead by one instruction)

SPT_CS_MODE_CTRL[DEBUG_MD] = 10b: Step to the next breakpoint mode.
SPT_CS_MODE_CTRL[DEBUG_MD] = 11b: Instruction jamming mode

The sequencer exits the DEBUG state if SPT_CS_MODE_CTRL[CS_IMM_DEBUG] is reset and SPT_CS_MODE_CTRL[CS_BKPT_DEBUG] is reset and if either SPT_CS_MODE_CTRL[CS_SYS_DEBUG] field is reset or system debug signal is low.

It should be noted that the breakpoints locations should not be set on the first program location (which corresponds to the value stored in register SPT_CS_PG_ST_ADDR).

45.9.2.7.2 Instruction Jamming mode

The instruction jamming feature is provided in the command sequencer to directly fill instructions through SW. This feature can be used during debugging to arbitrarily insert commands in a program flow. For example, a PDMA command could be inserted at a certain location in the program flow to DMA out data from the operand RAM or twiddle RAM for analysis.

Instruction jamming can be used only in the DEBUG state. To use this feature SW can set SPT_CS_MODE_CTRL[DEBUG_MD] == 11b. It should then also put the required succeeding instruction in SPT_CS_JAM_INST0, SPT_CS_JAM_INST1, SPT_CS_JAM_INST2 and SPT_CS_JAM_INST3 registers (these 4 32-bit registers contain the 128-bit jamming instruction), and reset SPT_CS_STATUS0[JAM_OVR] by writing a ‘1’ to it . The sequencer then executes the inserted instruction and halts and SPT_CS_STATUS0[JAM_OVR] gets set. If SPT_CS_INTEN0[JAM_OVR_INTEN] is also set then an interrupt is issued when SPT_CS_STATUS0[JAM_OVR] gets set.

Note that the SYNC, LOOP, NEXT and JUMP instructions should not be used in jamming mode; these will cause a flag to be set indicating that an inadmissible instruction was fired and command sequencer goes to ASYNCSTOP state. Similarly use of invalid opcodes will cause a flag to be set and sequencer goes to ASYNCSTOP state. It is also not advisable to run the WAIT instruction in the jam mode since it will alter the state of the command sequencer fsm. WAIT instruction is intended only to be used for diagnostic purposes i.e. to verify that the command sequencer fsm correctly goes to the WAIT state on execution of WAIT command and to check the working of WAIT module.

Note that asynchronous operation is not supported in the instruction jamming mode. An asynchronous PDMA instruction will be treated as a synchronous PDMA instruction. If a command error occurs then the command sequencer goes to the ASYNCSTOP state.

45.9.2.7.3 Stepping mode

In the stepping modes the sequencer can execute instructions one-by-one under SW control or continuously execute instructions till it reaches pre-defined breakpoints. SW can set four breakpoints at specific locations (corresponding to the absolute system memory address) in the program flow. When the command sequencer is in the DEBUG state and SPT_CS_MODE_CTRL[DEBUG_MD] == 01b and SPT_CS_STATUS0[STEP_ONCE_OVR] is reset (by writing a '1' to it) the sequencer executes one instruction and then halts and SPT_CS_STATUS0[STEP_ONCE_OVR] is set. If SPT_CS_INTEN0[STEP_ONCE_INTEN] is also set then an interrupt is issued when CS_STATUS0[STEP_ONCE_OVR] gets set.

Similarly if SPT_CS_MODE_CTRL[DEBUG_MD] == 10b and SPT_CS_STATUS0[STEP_JUMP_OVR] is reset (by writing a '1' to it) then the sequencer keeps executing instructions till it reaches the next enabled breakpoint and halts, and if there is no ongoing asynchronous PDMA instruction the SPT_CS_STATUS0[STEP_JUMP_OVR] field is set. If SPT_CS_INTEN0[STEP_JUMP_INTEN] is also set then an interrupt is issued when CS_STATUS0[STEP_JUMP_OVR] gets set.

It should be noted that in the 'step to next breakpoint' mode the sequencer compares the address of the present instruction being executed with the addresses of the four breakpoints and if any of them matches (and the corresponding breakpoint is enabled) the program execution breaks at the end of the instruction.

45.9.3 SPT Acquisition

45.9.3.1 Introduction

The data acquisition block receives maximum 14-bit samples from the ADCs or MIPICSI2, cuts out the samples in the acquisition window and performs on-the fly mathematical operations with these samples. The 16-bit results are forwarded to the SDMA (Sample DMA). Acquisition starts when it is enabled and triggered by the CTE (Cross Timing Engine) for ADC samples.

The following figure shows the block diagram of the Acquisition Block. ACQ_OPS stands for Acquisition operations module.

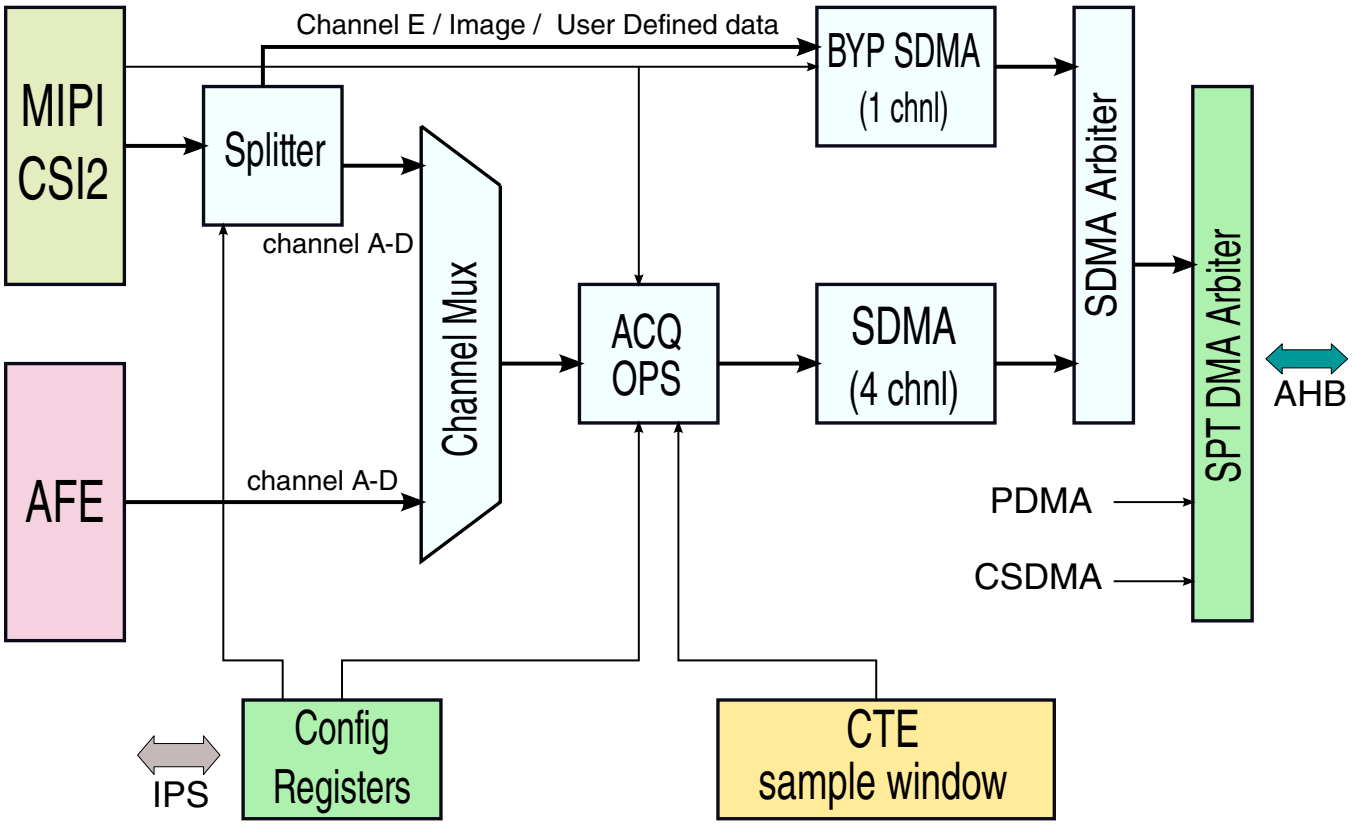


Figure 45-9. Acquisition Block

The ADC module provides 14-bit data which will be qualified by a data valid pulse. The data valid pulses for the enabled ADC channels can appear either concurrently or sequentially. For concurrent sampling mode, the data valid pulses for all ADC channels appear simultaneously while in sequential sampling mode, they appear in a staggered fashion with the order of staggering maintained for all 8 channels.

45.9.3.2 MIPICSI2 Interface

The register field ACQ_GBL_CTRL_0[CSI_ADC_SEL] is used to decide the source of the data to be acquired from. Acquisition receives frame start, chirp start and acquisition window from MIPICSI2, similar to the signals received from CTE in case of ADC as source. CTE phase inverse signal should be tied to 0. Along with the 24 bit input data, it also gives the data valid and data type signal to acquisition block. The data type can be changed within a frame but it should remain the same throughout a chirp. When combining Embedded/User defined data type with RGB/YUV data type, SPT_ACQ_GBL_CTRL_0[INF_SMPLS] bit needs to be set. For the 4 channel mode,

20MSPS is the maximum data rate supported in each channel for RAW8/10/12. RAW14 does not support 20MSPS. SPT receives data for all the channels from MIPICSI2 and then discards the data corresponding to the disabled channels.

The splitter decodes the incoming data based on the register ACQ_CSI_CTRL. If the data type is RAW 8/10/12/14, it is assigned channel numbers and stored in the 4chnl SDMA after arithmetic operations are performed on it. If the fifth channel is configured, the data is sign extended and then MSB aligned and stored in Bypass SDMA. If the data type is RGB/YUV/Embedded/User defined, it is directly stored in the 1 channel Bypass SDMA. RGB and YUV-10bit data write in two 16-bit locations of Bypass SDMA each valid cycle, Embedded and User defined data write in one 16-bit location every two valid cycle, while the rest of the formats one 16 bit location is written every valid cycle. Hence, if the valid cycles are in odd numbers (infinite samples in chirp) for embedded and user defined data, the last sample will be missed.

The possible sequences as programmed for the RAW data formats to be given to acquisition are shown in figures below for both with and without fifth channel. The (*) marked data are repetitive samples which are not stored in the SDMA. The data rate given for each sequence is the maximum possible data rate.

A-D: 10MSps E: 10 MSps	A-D: 5MSps E: 10 MSps	A-D: 2.5MSps E: 10 MSps	A-D: 6.67MSps E: 13.33 MSps	A-D: 5MSps E: 10 MSps
CH_A0	CH_A0	CH_A0	CH_A0	CH_A0.im
CH_E0	CH_E0	CH_E0	CH_B0	CH_A0.re
CH_B0	CH_B0	CH_B0	CH_C0	CH_E0.
CH_E0*	CH_E0*	CH_E1	CH_D0	CH_E0*
CH_C0	CH_C0	CH_C0	CH_E0	CH_B0.im
CH_E0*	CH_E1	CH_E2	CH_E1	CH_B0.re
CH_D0	CH_D0	CH_D0	CH_A1	CH_E0*
CH_E0*	CH_E1*	CH_E3	CH_B1	CH_E0*
CH_A1	CH_A1	CH_A1	CH_C1	CH_C0.im
CH_E1	CH_E2	CH_E4	CH_D1	CH_C0.re
CH_B1	CH_B1	CH_B1	CH_E2	CH_E1.
CH_E1*	CH_E2*	CH_E5	CH_E3	CH_E1*
CH_C1	CH_C1	CH_C1	CH_A2	CH_D0.im
CH_E1*	CH_E3	CH_E6	CH_B2	CH_D0.re
CH_D1	CH_D1	CH_D1	CH_C2	CH_E1*
CH_E1*	CH_E3*	CH_E7	CH_D2	CH_E1*
CH_A2	CH_A2	CH_A2	CH_E4	CH_A1.im
CH_E2	CH_E4	CH_E8	CH_E5	CH_A1.re
CH_B2	CH_B2	CH_B2	CH_A3	CH_E2.
CH_E2*	CH_E4*	CH_E9	CH_B3	CH_E2*
CH_C2	CH_C2	CH_C2	CH_C3	CH_B1.im
CH_E2*	CH_E5	CH_E10	CH_D3	CH_B1.re
CH_D2	CH_D2	CH_D2	CH_E6	CH_E2*
CH_E2*	CH_E5*	CH_E11	CH_E7	CH_E2*

Figure 45-10. Input data sequence with ChannelE

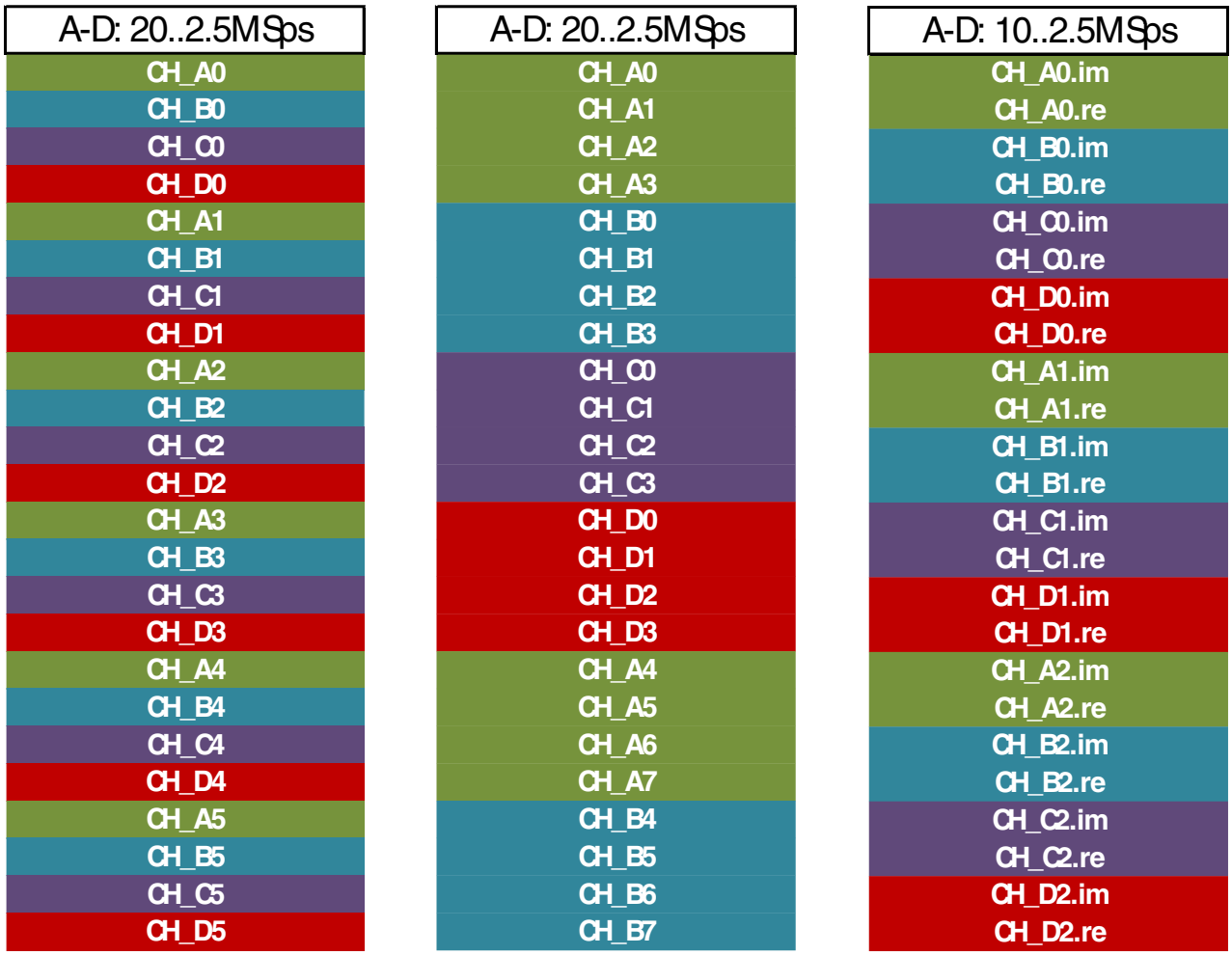


Figure 45-11. Input data sequence without ChannelE

45.9.3.3 Channel Select and Re-order

The channels selector switches between MIPICSI2 and the integrated ADCs. Each channel can be enabled or disabled for further processing by a mask, which is programmed with the configuration registers. Refer to the ACQ_GBL_CTRL_0[CH_x_SEL] fields for more information. When a channel is disabled its samples are invalid.

In order to simplify PCB routing it is possible to swap the channel numbers of ADCs, e.g. ADC_B channel becomes ADC_A and so on. Only a swap of two neighboring channels and a channel reverse swap is supported. Swapping does not apply for samples from MIPICSI2. If programmed wrongly, the channel will be treated as invalid. The figure and table below provide information on the swapping operation for both cases of 4ADC channels and 8ADC channels.

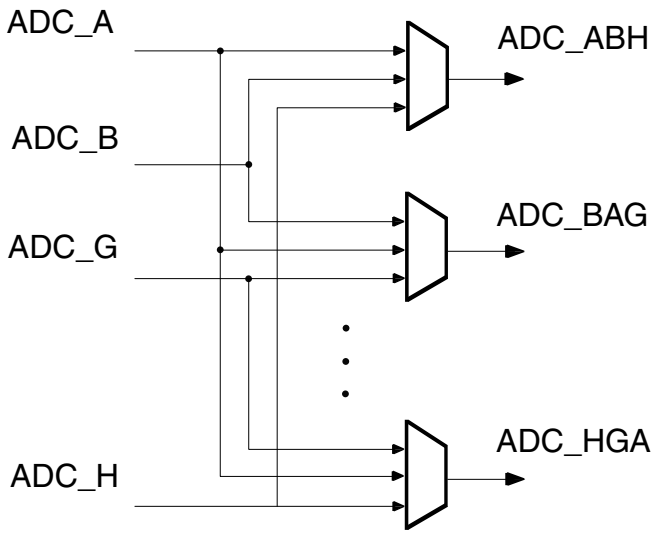


Figure 45-12. ADC Channel Multiplexer

Swap	Output channel A	B	C	D
Input channel A	X	X		X
Input channel B	X	X	X	
Input channel C		X	X	X
Input channel D	X		X	X

Swap	Output channel A	B	C	D	E	F	G	H
Input channel A	X	X						X
Input channel B	X	X					X	
Input channel C			X	X		X		
Input channel D			X	X	X			
Input channel E				X	X	X		
Input channel F			X		X	X		
Input channel G		X					X	X
Input channel H	X						X	X

Figure 45-13. ADC Channel Swap

45.9.3.4 Acquisition Window

The acquisition window is defined by the window start signal of the CTE and a programmable length in sampling clock pulses in a chirp (default value is 256). Acquisition starts with the rising edge of this window. It stops when either sample counter is exceeded or with the falling edge of the window signal provided by the CTE; whichever is earlier. When MIPICSI2 interface is enabled, acquisition receives frame start, chirp start, acquisition window signals from MIPICSI2 instead of CTE.Multiple Acquisition windows within a single chirp is not supported.

If window signal from CTE/MIPICSI2 deasserts before programmed number of samples have been acquired, it is possible (in case of sequential sampling) that some channels have provided data, while others were yet to provide data. In such cases, acquisition block appends junk data to complete data acquisition from all channels and proceeds to transfer this data to SRAM via SDMA. An exception will be generated in such cases.

During Initialization, acquisition block must be enabled (by setting SPT_GBL_CTRL[ACQ_EN]), before arrival of RFS (radar frame start) from CTE. This should be followed by assertion of window signal from CTE, thus initiating acquisition of radar samples.

NOTE

User must ensure that the minimum time between de-assertion of acquisition window at chirp boundary and subsequent chirp/ frame start (programmable via CTE) must be at least 1us for proper operation of acquisition block. Similarly, the time between deassertion of chirp and subsequent assertion of its acquisition window must be greater than the sum of 5 AFE/ MIPICSI2 interface clock cycles and 5 AHB interface clock cycles.

NOTE

When acquisition window does not appear after a chirp (i.e the chirp is followed by another chirp with any acquisition window) or there are valid samples without acquisition window, such a chirp is known to be an empty chirp. An empty chirp results in the following:

- Chirp Counter does not increment for an empty chirp
- In 2D Mode, SDMA Buffer address does not increment by line length after an empty chirp
- An empty chirp is not counted in the Buffer Length count which is used to reset the SDMA Base Address.

NOTE

Please ensure that the width of RFS/RCS pulses going towards Acquisition are greater than or equal to 2 MIPICSI2 clock periods for MIPICSI2 interface and 2 AHB clock periods for AFE interface.

NOTE

If infinite number of samples are programmed for an interleaved format read with AFE interface or Embedded/User defined data type in MIPICSI2 interface, then acquisition

window should go low when even number of samples have been acquired to avoid data loss.

45.9.3.5 Acquisition Operations

The mathematical operations performed on the incoming 14-bit signed data sample are: min, max, toggle check, sum, offset compensation and phase inverse compensation. The results of min, max, toggle check and sum are stored in registers. Offset and phase inverse compensation change the values of the samples. Before offset and phase inverse compensation, the incoming samples are left shifted by 1 bit and sign extended to 16 bits. The values for offset compensation are programmed by configuration registers (ACQ_CTRL0/1/2/3). The following figure provides implementation details.

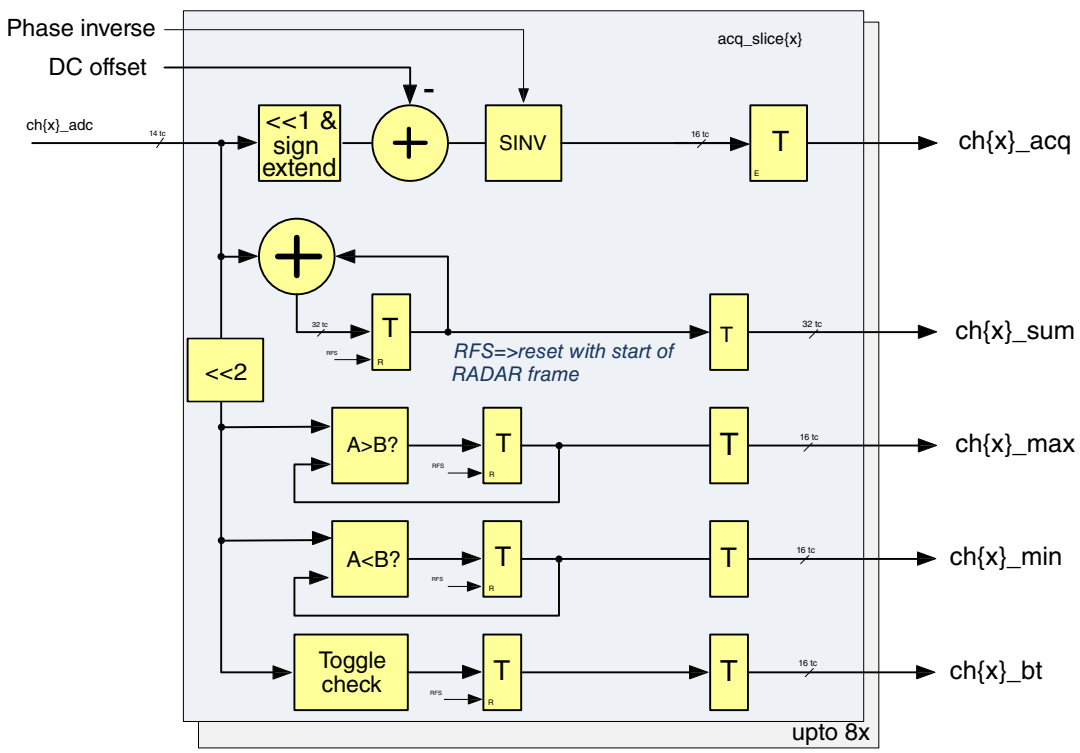


Figure 45-14. Acquisition Operations

Min, max, toggle check and sum operations are calculated on all valid samples during the acquisition window only. These operations are performed over the entire RADAR frame. With the start of the RADAR frame, indicated by RFS, the accumulating registers are cleared. Phase inverse value and the offset are constant for single acquisition window. The offset can be different for each channel but the phase inverse value is common to all channels.

45.9.3.5.1 Min, Max and Sum Calculation

The 'Min' operation (refer ch{x}_min in the above figure) looks for the global minimum value within the acquisition windows over the entire frame. Similarly 'Max' operation (refer ch{x}_max in the above figure) looks for the global maximum value. The 'Sum' operation (refer ch{x}_sum in the above figure) accumulates the samples during acquisition window to calculate sum of all samples over the entire frame. The results of these operations are signed. The result from the 'Sum' operation can be used by the CPU for average calculation.

The results of these operations are stored in the acquisition status registers ranging from ACQ_STATUS_x0 to ACQ_STATUS_x1, where 'x' represents corresponding ADC channel. The results for "Min" and "Max" operations are MSB aligned and stored in a 16 bit field. The unused LSB's are padded with zero's. These statistics are valid to be sampled only after end of frame event.

45.9.3.5.2 Toggle Check

The toggle check is a reliability feature and shall check that each bit can change. For this purpose, each single bit is observed during acquisition window, and if toggled at least once (i.e. one rising and one falling edge), then the corresponding bit position is set to ‘1’. The result of this operation is MSB aligned and stored in a 16 bit field in the acquisition status register ACQ_STATUS_x2, where 'x' represents corresponding ADC channel. The unused LSB bits DO NOT TOGGLE and are padded with zeros. These statistics are valid to be sampled only after end of frame event.

45.9.3.5.3 Offset and Phase Inverse Compensation

Phase Inverse compensation is a specific operation to compensate a pi phase shift by sign inversion. The phase inverse signal is provided by CTE. If this signal is logic '0', no phase inversion is performed, while logic '1' leads to phase inversion operation. If the incoming samples are from MIPICSI2, this signal should be logic '0'.

Offset compensation logic corrects the ADC offset values, depending on the result of the mean value of the previous cycles. These offset values are programmed using configuration registers (ACQ_CTRL0/1/2/3). This offset value is subtracted from the left-shifted sample value as shown in figure "Acquisition Operations" above.

45.9.3.6 Sample DMA

The SDMA is responsible for merging ADC samples in to memory words and to arrange the data into packets.

There are two sample DMAs in acquisition:

- 4 channel SDMA
- 1 channel Bypass SDMA

4 channel SDMA stores the shifted data from the incoming channels, while the Bypass SDMA stores image data and the fifth channel calibration data. Only when the fifth channel is on, both the SDMAs have to be configured appropriately. If its the ADC or the RAW data MIPICSI2 interface only the 4 channel SDMA should be configured and if image data is being received only the Bypass SDMA should be configured. Both the SDMAs are connected to a SDMA arbiter, which processes the requests in round robin fashion and gives it to the SPT arbiter.

45.9.3.6.1 ADC Data Transfer

Four 16-bit ADC samples are used to build a 64-bit word for bus transfer to system RAM memory. The number of samples being transferred for each chirp is always equal to the configured number of samples in acquisition window multiplied by the number of enabled ADC channels.

Figure 45-15. ADC Data Transfer

45.9.3.6.2 Sample Interleaving

The samples of the ADC channels are interleaved channel-wise and collected in DMA buffer. The number of active channels and their order can be configured using ACQ_GBL_CTRL_0 register. Multiple data organization methods are available. This can be configured using SDMA_CTRL1[DATA_ORG_CFG] field.

Denoting the ADC channels with A,B,C,… and the sample count with A_x then following configurations are supported:

- Interleaved : $A_0B_0C_0D_0$, $A_1B_1C_1D_1$, ..., $A_{n-1}B_{n-1}C_{n-1}D_{n-1}$
- Tile 4 : $A_0A_1A_2A_3$, $B_0B_1B_2B_3$, ..., $D_{n-4}D_{n-3}D_{n-2}D_{n-1}$
- Tile 8 : $A_0A_1A_2A_3$, $A_4A_5A_6A_7$, $B_0B_1B_2B_3$, ..., $D_{n-4}D_{n-3}D_{n-2}D_{n-1}$

The interleaved mode can be used for SPT operations only when the number of channels enabled are greater than 4. For the case where the number of channels enabled is less than 5, they can only be used when SPE (Signal Processing Engine in CPU) needs to work on

the data, and not SPT. This is because interleaved mode with less than 5 channels leads to 4 ADC samples being packed together back to back. This particular form of data organization is not usable by Hardware Accelerator blocks within SPT.

The tile transfer determines the intermediate buffer size to bridge DMA latencies. To support Tile-8 mode, it has 8 channels x 8 words x 16 bits x 2 (double buffering) = 256 bytes.

ADC	A	B	C	D	E	F	G	H
case 1	1	1	1	1	1	1	1	1
case 2	1	1	1	1	0	0	0	0
case 3	1	1	0	1	0	0	0	0
case 4	1	0	0	1	1	1	0	0
case 5	1	1	0	0	0	0	0	0
case 6	0	0	0	0	0	1	0	0

word 1				word 2				word 3				word 4				word 5			
A1	B1	C1	D1	E1	F1	G1	H1	A2	B2	C2	D2	E2	F2	G2	H2	A3	B3	C3	D3
A1	B1	C1	D1	A2	B2	C2	D2	A3	B3	C3	D3	A4	B4	C4	D4	A5	B5	C5	D5
A1	B1	0	D1	A2	B2	0	D2	A3	B3	0	D3	A4	B4	0	D4	A5	B5	0	D5
A1	0	0	D1	E1	F1	0	0	A2	0	0	D2	E2	F2	0	0	A3	0	0	D3
A1	B1	0	0	A2	B2	0	0	A3	B3	0	0	A4	B4	0	0	A5	B5	0	0
0	F1	0	0	0	F2	0	0	0	F3	0	0	0	F4	0	0	0	F5	0	0

Figure 45-16. Interleaved storage ADC/MIPICSI2 Real data

ADC	A	B	C	D	E	F	G	H
case 1	1	1	1	1	1	1	1	1
case 2	1	1	1	1	0	0	0	0
case 3	1	1	0	1	0	0	0	0
case 4	1	0	0	1	1	1	0	0
case 5	1	1	0	0	0	0	0	0
case 6	0	0	0	0	0	1	0	0

word 1				word 2				word 3				word 4				word 5			
A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4	D1	D2	D3	D4	E1	E2	E3	E4
A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4	D1	D2	D3	D4	A5	A6	A7	A8
A1	A2	A3	A4	B1	B2	B3	B4	D1	D2	D3	D4	A5	A6	A7	A8	B5	B6	B7	B8
A1	A2	A3	A4	D1	D2	D3	D4	E1	E2	E3	E4	F1	F2	F3	F4	A5	A6	A7	A8
A1	A2	A3	A4	B1	B2	B3	B4	A5	A6	A7	A8	B5	B6	B7	B8	A9	A10	A11	A12
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20

Figure 45-17. Tile 4 Storage ADC/MIPICSI2 Real data

ADC	A	B	C	D	E	F	G	H
case 1	1	1	1	1	1	1	1	1
case 2	1	1	1	1	0	0	0	0
case 3	1	1	0	1	0	0	0	0
case 4	1	0	0	1	1	1	0	0
case 5	1	1	0	0	0	0	0	0
case 6	0	0	0	0	0	1	0	0

word 1				word 2				word 3				word 4				word 5			
A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	C1	C2	C3	C4
A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	C1	C2	C3	C4
A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	D1	D2	D3	D4
A1	A2	A3	A4	A5	A6	A7	A8	D1	D2	D3	D4	D5	D6	D7	D8	E1	E2	E3	E4
A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	A9	A10	A11	A12
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20

Figure 45-18. Tile 8 Storage ADC/MIPICSI2 Real data

ADC	A.im	A.re	B.im	B.re	C.im	C.re	D.im	D.re
case 1	1	1	1	1	1	1	1	1
case 2	1	1	1	1	0	0	0	0
case 3	1	1	0	0	0	0	0	0
case 4	0	0	0	0	1	1	1	1
case 5	1	1	1	1	1	1	1	1
case 6	0	0	0	0	0	0	1	1

Storage Type
Interleaved
Interleaved
Tile 4
Tile 4
Tile 8
Tile 8

word 1				word 2				word 3				word 4				word 5			
A1.im	A1.re	B1.im	B1.re	C1.im	C1.re	D1.im	D1.re	A2.im	A2.re	B2.im	B2.re	C2.im	C2.re	D2.im	D2.re	A3.im	A3.re	B3.im	B3.re
A1.im	A1.re	B1.im	B1.re	0	0	0	0	A2.im	A2.re	B2.im	B2.re	0	0	0	0	A3.im	A3.re	B3.im	B3.re
A1.im	A2.im	A3.im	A4.im	A1.re	A2.re	A3.re	A4.re	A5.im	A6.im	A7.im	A8.im	A5.re	A6.re	A7.re	A8.re	A9.im	A10.im	A11.im	A12.im
C1.im	C2.im	C3.im	C4.im	C1.re	C2.re	C3.re	C4.re	D1.im	D2.im	D3.im	D4.im	D1.re	D2.re	D3.re	D4.re	C5.im	C6.im	C7.im	C8.im
A1.im	A2.im	A3.im	A4.im	A5.im	A6.im	A7.im	A8.im	A1.re	A2.re	A3.re	A4.re	A5.re	A6.re	A7.re	A8.re	B1.im	B1.im	B1.im	B1.im
D1.im	D2.im	D3.im	D4.im	D5.im	D6.im	D7.im	D8.im	D1.re	D2.re	D3.re	D4.re	D5.re	D6.re	D7.re	D8.re	D9.im	D10.im	D11.im	D12.im

Figure 45-19. MIPICSI2 complex data storage

The image data from MIPICSI2 is stored in 64-bit words as given by the following figure.

Format	MSB																													LSB
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB888	8'b0								r0[7:0]								g0[7:0]								b0[7:0]							
	8'b0								r1[7:0]								g1[7:0]								b1[7:0]							
YUV422-8	Y0[7:0]								U0[7:0]								Y1[7:0]								V0[7:0]							
	Y2[7:0]								U2[7:0]								Y3[7:0]								V2[7:0]							
ED/UD	data0[7:0]								data1[7:0]								data2[7:0]								data3[7:0]							
	data4[7:0]								data5[7:0]								data6[7:0]								data7[7:0]							
YUV-10bit	Y0[9:0]										6'b0						U0[9:0]										6'b0					
	Y1[9:0]										6'b0						V0[9:0]										6'b0					
RGB565	8'b0										R0[4:0]					3b'0	G0[5:0]						2b'0				B0[4:0]				3b'0	
	8'b0										R1[4:0]					3b'0	G1[5:0]						2b'0				B1[4:0]				3b'0	

Figure 45-20. Image data storage

45.9.3.6.3 SDMA Data Transfer

The data is transferred to system memory (system RAM/TCM, preferably a separate bank which is not being used by any other Master like for ex: Flexray). The transfer to main memory is performed by the SDMA module, depending on its configuration. The DMA operation is scheduled once and runs autonomously when enough samples are available in buffer.

The data transfer is triggered, when one of the two buffers of either of the SDMAs has been filled. The burst size is dependent on the SDMA data organization selected and the number of channels enabled. Refer DMA Arbitration for details on various burst sizes available. The final burst after the acquisition window may be residual burst, with less data. The ADC sample buffer is flushed with the last burst of a chirp. Whenever a burst is done, data from the other SDMA is transferred in case atleast one buffer of each SDMA is full. The bypass SDMA always transfers the 64x16bits buffer at once, so if the number of samples programmed are not a multiple of 64 (multiple of 128 in case of Embedded/ User Defined), some junk data will be appended in the last burst. Similarly, when 4 or less channels are enabled for MIPICSI2 all modes or ADC interleaved mode, if the number of samples is not a multiple of 16, some junk data will be transfered in the end.

45.9.3.6.4 SDMA Addressing Modes

The SDMA supports linear write (1D) and orthogonal (2D) write operations. This can be configured using SDMA_CTRL1[TRF_MOD] bit.

With linear write mode, the data is transferred consecutively to the target memory, i.e. each ADC data packet follows the other. The target address is auto-incremented with the next transfer by the number of previous transferred data words. The target address is reset to the start address with an active edge of the frame start signal.

With orthogonal write mode, the data is transferred to the target memory while the configured line length is not exceeded, otherwise the data is dropped. The address is auto-incremented like in linear write mode until the next active edge of chirp start signal (RCS). At this moment, the next target address is calculated as sum of previous start address and the line length. Line length can be configured using SDMA_CTRL1[LINE_LEN] field. The target address is also reset to the original start address with an active edge of the frame start signal (RFS). Any unused memory is not filled.

The Buffer Length field programmed by SDMA_CTRL1[SDMA_BUF_LEN] provides the number of chirps for which samples are stored in System RAM back to back; after which target address is reset to start address from next successive chirp onwards. This happens regardless of linear/orthogonal mode. However, a frame start signal always causes target address to reset to start address. Refer to SDMA_CTRL1[SDMA_BUF_LEN] for more details.

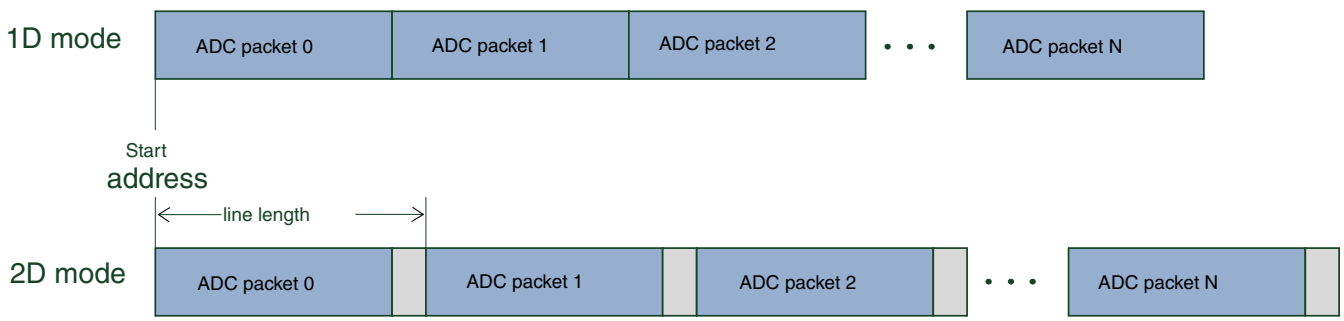


Figure 45-21. SDMA Modes

45.9.4 Operand RAM

45.9.4.1 Introduction

The operand RAM stores the operands for operations like FFT, Histogram, Programmable DMA (PDMA). The operand RAM is divided into 4 banks. Each bank consists of 8 slices. Each slice has a width of 48 bits (+4 parity bits). Each bank can either have a read access or a write access every clock cycle.

The operand RAM configuration is :

8 slices of (48 bits +4 parity bits) x 512 depth per bank. There are 4 such banks.

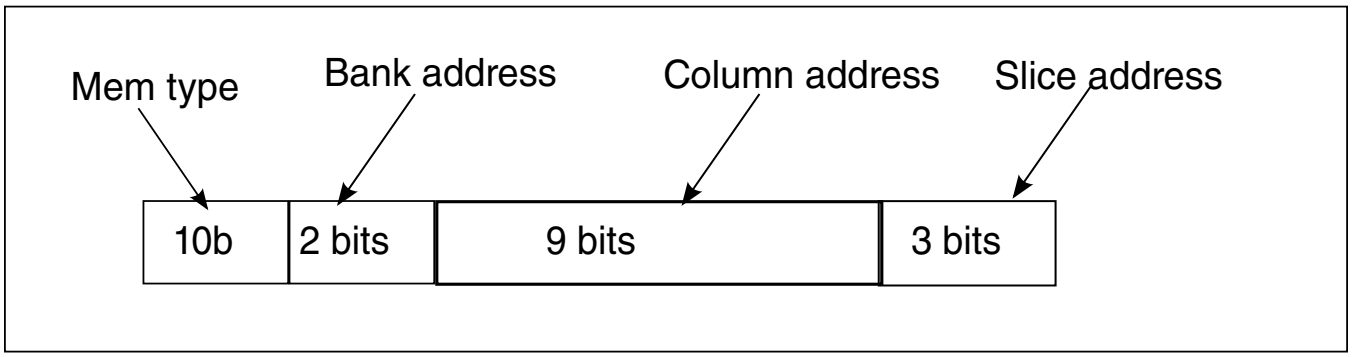


Figure 45-22. Operand RAM addressing bits

45.9.4.2 Operand RAM : Error Detection

Two parity bits are provided for every 24 bits. Parity bit 0 contains the XORed value of all the even bits and parity bit 1 contains the XORed values of all the odd bits. The arrangement of data and parity bits are shown in the figure below.

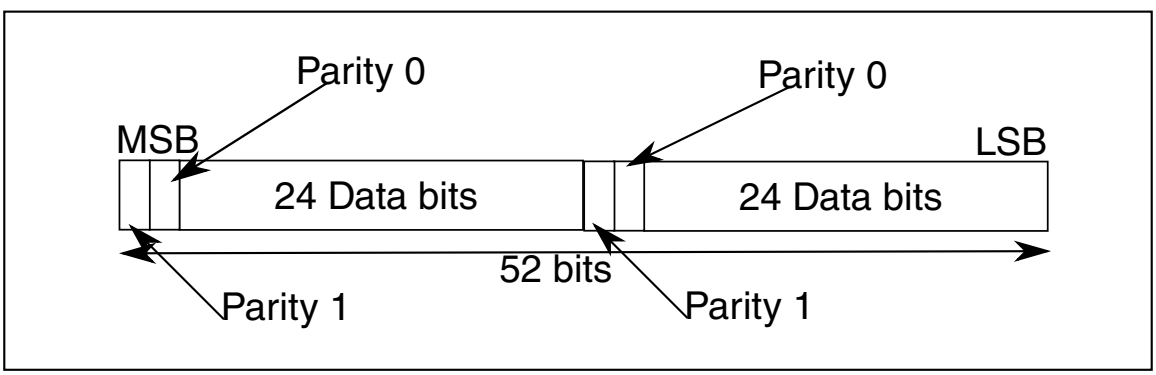


Figure 45-23. Data and parity

45.9.4.3 Operand RAM accesses

The operand RAM is accessed by all the hardware accelerators: COPY, MAXS,HIST, FFT, VMT, GET, SET and ADD instructions. All the hardware accelerators are allowed to have one read and one write transaction on the operand RAM provided they are not on the same bank. The PDMA is allowed to have one read or write transaction to the operand RAM. For every address accessed, the operand RAM controller provides the data for 8 consecutive locations. Slice unaligned accesses are not allowed. For every address coming to the SPT RAM controller, data is read from/written to all the 8 slices in a bank. Word enables are provided to selectively write data to any 24 bit word.

Every bank can have either a read or a write transaction at one time. Simultaneous reads and writes to one bank are not allowed. In case a hardware accelerator requests for reads and writes in the same bank, writes will be given a priority and "wait" will be asserted for the read transaction. In case there is a contention on any one bank between the hardware accelerators and the PDMA, the PDMA is given a priority and "wait " is asserted for the hardware accelerators.

Figure 45-24 shows the locations accessed with an aligned slice address.

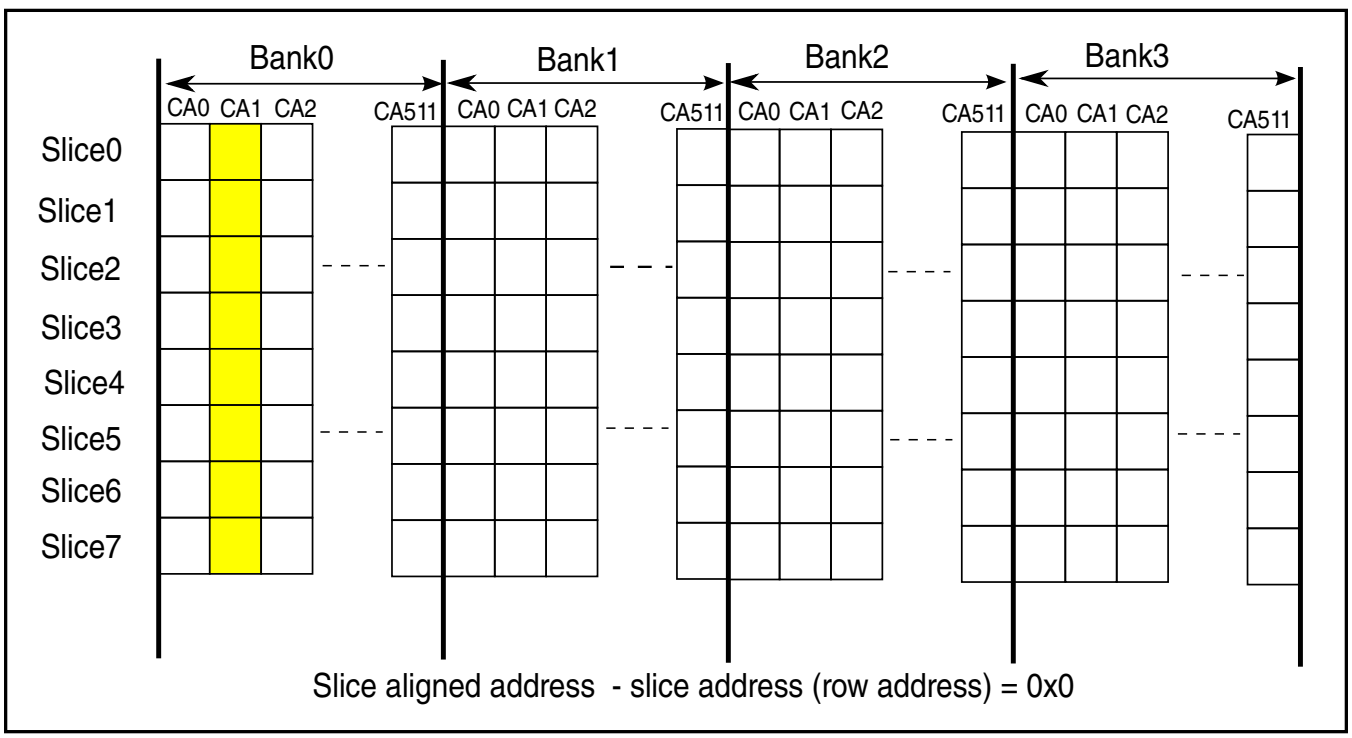


Figure 45-24. Slice aligned access

45.9.5 Twiddle RAM

45.9.5.1 Introduction

The twiddle RAM holds constants like coefficients, which are used during some operations. It is also organized in slices of 8 to enable parallel access to 8 coefficients simultaneously and can be initialized with DMA operations. Operations can use the twiddle RAM as source and destination, but concurrent read and write is not supported. All hardware accelerators and DMA can access the twiddle RAM. However, at one time only one interface can be active.

The configuration of the twiddle RAM is :

8 slices of 36 bits X 512 depth. Each location has 32 bits of data and 4 bits of parity.

It is possible to read 8 different locations, one in each slice. Therefore, 8 address buses are required from each of the modules. The addressing bits for the twiddle RAM is shown in Figure 45-25.

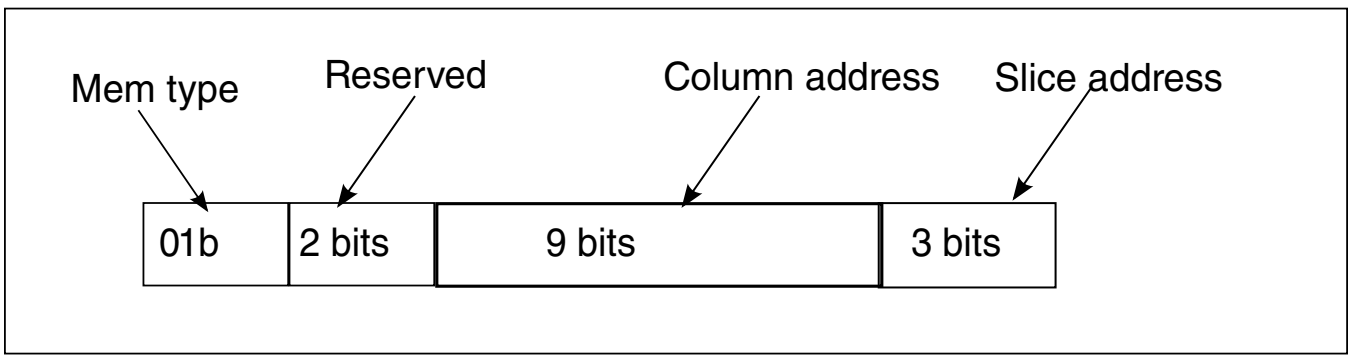


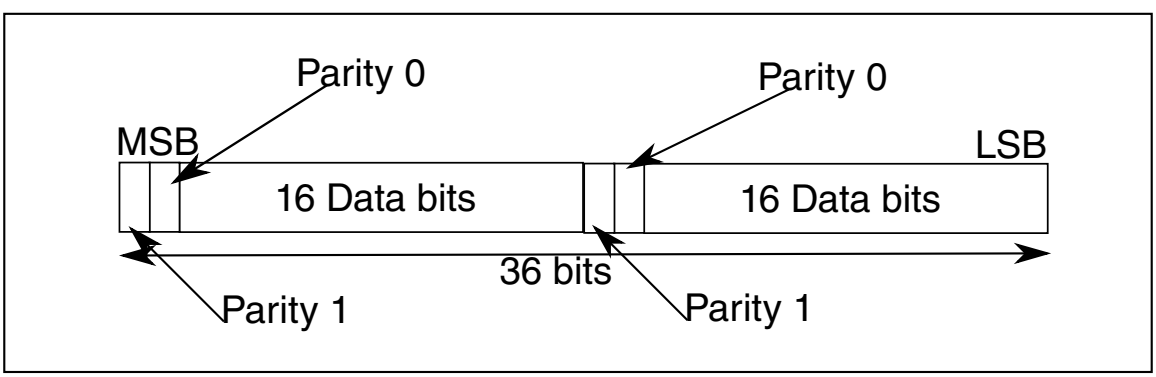
Figure 45-25. Twiddle RAM address bitwidths

Since there is only one bank in twiddle RAM, the two bank bits of address are don't care.

45.9.5.2 Twiddle RAM : Error Detection

Two parity bits are provided for every 16 data bits.Parity bit 0 contains the XORed value of all the even bits and parity bit 1 contains the XORed values of all the odd bits. The arrangement of data and parity bits are shown in the Figure 45-26.

Figure 45-26. Data and parity



45.9.5.3 Twiddle RAM Organization and Twiddle Generation

Twiddles are organized as 8 slices by N/8 entries as shown below:

Col. addr.	CA0	CA1	CA2	CA3	CA4	...	CA_N/8-1
slice0	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice1	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice2	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice3	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice4	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice5	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice6	tw1	tw2	tw3	tw4	tw5	...	twN/8
slice7	tw1	tw2	tw3	tw4	tw5	...	twN/8

For example, for FFTs of size 512, there are 8 entries for each twiddle factor and a total of N/8 or 64 unique twiddle factors.

Twiddle factors are generated by using the formula Equation 13 on page 1982

$$W_N^k = e^{-j2\pi k/N}$$

Equation 13. Formula for generation of Twiddle Factor

Here N is the FFT size.

To generate twiddle factors, the following pseudo code can be used:

```
<codeblock>
for  n = 1 to FFT_Length / 8
    Real = int_scale * cos(-2 * pi * n / FFT_Length);
    Imaginary = int_scale * sin(-2 * pi * n / FFT_Length);
The Real and Imaginary parts of these equations can then be rounded to 16 bit integer values
and then placed into a 32 bit entry
</codeblock>
```

The user need only define twiddles for one quadrant (- Imaginary, + Real) and the SPT logic determines other needed twiddles by real / imaginary swaps and sign inversions. The SPT defines this real / imaginary swap and sign inversion process as quadrature extension.

As an example, consider the following twiddle factors for FFT length 32. First, a text style listing the 8 copies for 32 / 8 = 4 total unique twiddles is shown below:

```
#Twiddle RAM values for FFT length 32
0x4000: 32137 +i -6393
0x4001: 32137 +i -6393
0x4002: 32137 +i -6393
0x4003: 32137 +i -6393
0x4004: 32137 +i -6393
0x4005: 32137 +i -6393
0x4006: 32137 +i -6393
0x4007: 32137 +i -6393
0x4008: 30273 +i -12539
0x4009: 30273 +i -12539
0x400a: 30273 +i -12539
0x400b: 30273 +i -12539
0x400c: 30273 +i -12539
0x400d: 30273 +i -12539
```

```
0x400e: 30273 +i -12539
0x400f: 30273 +i -12539
0x4010: 27245 +i -18204
0x4011: 27245 +i -18204
0x4012: 27245 +i -18204
0x4013: 27245 +i -18204
0x4014: 27245 +i -18204
0x4015: 27245 +i -18204
0x4016: 27245 +i -18204
0x4017: 27245 +i -18204
0x4018: 23170 +i -23170
0x4019: 23170 +i -23170
0x401a: 23170 +i -23170
0x401b: 23170 +i -23170
0x401c: 23170 +i -23170
0x401d: 23170 +i -23170
0x401e: 23170 +i -23170
0x401f: 23170 +i -23170
```

Next, a C-style header file entry for the same FFT length of 32 is shown below:

```
/*Twiddle RAM values for FFT length 32*/
/*for use with 16bit complex PMDA transfer*/
/*tw[k].im, tw[k].re, tw[k+1].im, tw[k+1].re*/
const unsigned long fft_twd32[] = {
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xe7077d89,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xcf057641,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xb8e46a6d,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82,
    0xa57e5a82
};
```

45.9.6 Work register

45.9.6.1 Introduction

Certain SPT operations need single values for calculation (such as coefficients) or generate single output results (such as sum). Such values can be saved in result/operand registers.

A total of 48 such registers are available which can be directly accessed by the CPU via the peripheral interface. All Work Registers (R0 to R47) are 32-bit accessible only. The work register can also be used with specific commands to extract or initialize specific Operand RAM or Twiddle RAM cells.

The addressing bits for Work registers is shown in the figure below.

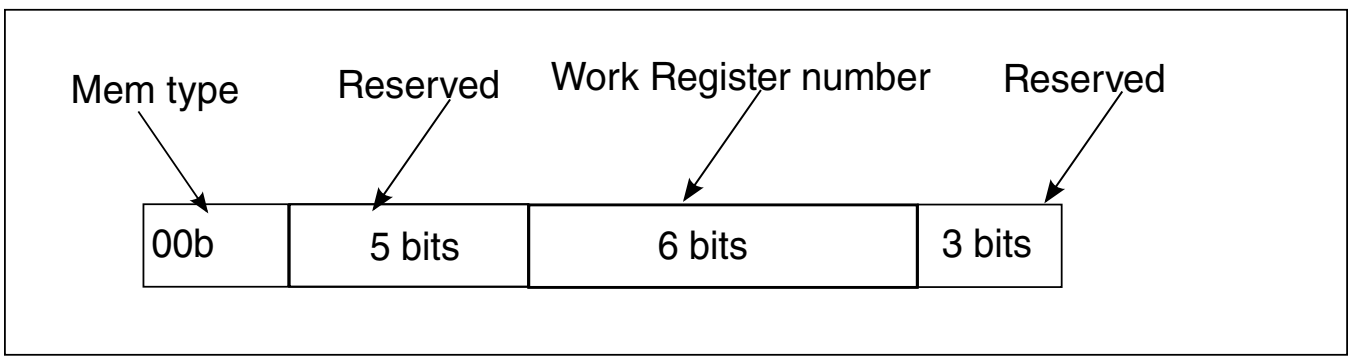


Figure 45-27. Work Register addressing bits

45.9.6.2 Work Register accesses

Each work register is a 48 bit register which is accessible to the SPT internal modules as well as the CPU and external parties. The CPU and external modules can access the work registers via the peripheral interface.

The following points define the behavior of the accesses to work registers.

1. Every work register contains a 48 bit data (24 bit real and 24 bit imaginary)
   a. Peripheral side view of each register are two consecutive registers with 24 bit value in each (rest 8 bit are reserved)
2. Writes have two modes
   a. Writes locked mode - Writes are locked for either CPU or SPT
   b. Writes unlocked mode - Both CPU and SPT can write into the register. Coherency has to be tackled by the software. In case of contention while writing

data in unlocked mode, SPT writes will be given a priority and the CPU write will be lost.

3. Two control bits are provided for each work register to define whether a work register is locked and for which party (SPT/CPU). WR_CTRL_x_REGs provide these control bits.
4. Reads are free from both the interfaces (SPT/CPU)
   a. Reads by the CPU via the peripheral interface assume that the data is stable (CPU is responsible to read the data when it is stable). CPU can use handshaking mechanisms to ensure that stable data is read.
5. Writes by the CPU via the peripheral interface will be written to the work registers after a delay - required for clock domain crossing. There wont be any wait assertion for the first write but wait will only be asserted if next read or write happens within 3 IPG + 2 SPT clock cycles
6. In case CPU writes to a work register locked for SPT, a transfer error is generated.
7. In case SPT writes to a work register locked for CPU, MEM_ERROR_REG[SPT_LCK_VIOL] bit is set and an interrupt can be generated if enabled.

It is recommended to use the writes locked mode of the work registers to ensure that there is no contention.

NOTE

The assumption here is that the peripheral clock frequency is slower than the frequency at which the SPT works.

NOTE

When SPT is disabled using SPT_GBL_CTRL[SPT_EN] and then enabled again, the values of the Work Registers are indeterminate. To ensure that the Work Registers are reset the user must write 0x0 to SPT_WR_R0_RE before enabling SPT. Other than for the purpose of resetting the work registers they should only be written when SPT_EN=1.

NOTE

When the SPT executes a STOP command at the end of a command sequence, the work register values are not deterministic until they are re-initialized. The CPU should not rely on access to the work registers between a STOP command and the launching of a new command sequence, instead the WAIT and EVT commands should be used to manage the timing of accesses when both SPT and CPU must interact with work registers during command sequence execution.

45.9.7 FFT core

45.9.7.1 Introduction

The FFT module implements two Radix4 kernels. Both Radix4 operations can only be used in the same context, e.g. to perform the Fast Fourier Transform (FFT), Inverse FFT, Scalar Product (SCP), Finite Impulse Response Filtering (FIR) or Windowing (WIN) operations. The kernel operation is auto-repeated for the operand length defined by the instruction. A complete FFT is built from several Radix4 or Radix2 rounds, each of which need a command.

The FFT module shall perform 2^N real and complex FFTs with N ranging from 3 to 12.

45.9.7.2 Features

- Automated operation for single rounds
- FFT and Inverse FFT basic operations
  - Radix4 butterfly and twiddle multiplication
  - Radix2 butterfly and twiddle multiplication
- Windowing for Pre and Post multiplication with coefficients
  - Constant, full vector and partial vector
- Twiddle handling
  - Offset/modulo address calculation for twiddles
  - Reduced twiddle storage
  - Sub-sampling of twiddles possible
- Scaling
  - Fixed result scaling
  - Input scaling with selectable gain
- Advanced calculation
  - FIR filter and Scalar Product operation
  - Real number FFT support
  - Combination of many small FFTs with single operation
- Reordering of input operands

45.9.7.3 Input Data

The input operand vector consists of real or complex values, stored in the operand RAM. For FFT operation the vector length N must be a power of two, greater than or equal to 8, and for Windowing operation, Scalar Product or FIR filtering the vector length should be a multiple of 8. Twiddles and multiplication coefficient vectors are fetched from twiddle RAM. A single multiplication coefficient can also be loaded from work registers or can be given as part of the command (immediate data).

45.9.7.4 Output Data

The results are stored in operand RAM.

45.9.7.5 Kernel Operations

A single Radix4 kernel may perform the following operations:

- Radix2 with two operands, performed two times in a single step using a single butterfly. Thus the two butterflies process eight input operands in each clock cycle. This operation is performed using the RDX2 or IRDX2 command.
- Radix4 with four operands, performed by a single butterfly. Thus the two butterflies in the kernel can process eight input operands each clock cycle. This operation is performed using the RDX4 or IRDX4 command.
- Windowing with four operands, by a single butterfly. This is essentially a complex multiplication operation. Thus the two butterflies can perform eight complex multiplications (or windowing operations) each clock cycle. This operation is performed using the WIN command. Window operations may precede or follow Radix rounds.
- Multiply/Accumulate with four operands, using a single butterfly. Thus the two butterflies can perform eight complex multiplications and the products are added together. This is similar to the operation of an FIR filter with eight taps.
- Multiply-add operation using a single or two butterflies. This operation is used to calculate the scalar product.

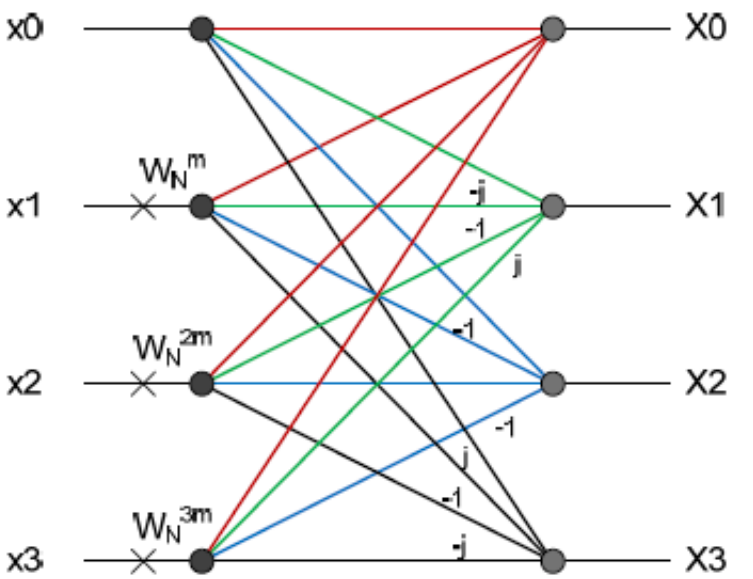
The source operands and the results are stored in the operand RAM, and the twiddles are read from twiddle RAM. The round counter in the Radix2 or Radix4 instruction determines the addressing scheme for source operands and the results as well as the read twiddles. Results are stored in way making concatenation of rounds easy. For this purpose, the Radix operation reads from and writes to same addresses, preferably at different banks to ensure full performance. If read and write were done on the same bank, then the performance would be affected; it would be slower by a factor of two.

The twiddle coefficients for RDX4, RDX2, IRDX4 and IRDX2 commands are stored in twiddle RAM.

The coefficients for windowing and multiply/accumulate and multiply-add operations are also stored in twiddle RAM. Additionally, windowing operation can use constant coefficients from the command word itself (immediate data) or a work register.

45.9.7.6 Principle of Operation

A special symbol is used for the Radix4 operation which is shown in the following picture. The operation consists of complex multiplications and a butterfly addition.



$$\begin{pmatrix} X0 \\ X1 \\ X2 \\ X3 \end{pmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{pmatrix} 1x0 \\ W_N^m x1 \\ W_N^{2m} x2 \\ W_N^{3m} x3 \end{pmatrix}$$

Figure 45-28. Radix4 Butterfly

The complete FFT is split into steps and rounds of Radix operations. A single command performs one round of selected operation, consisting of multiple Radix operations per round. A $N=4^k$ FFT (k=2,3,4,5,6) is completed after 4^{k-1} Radix operations per round and k rounds. Since each step executes two Radix operations, only $4^{k-1}/2$ steps are required to finish one round.

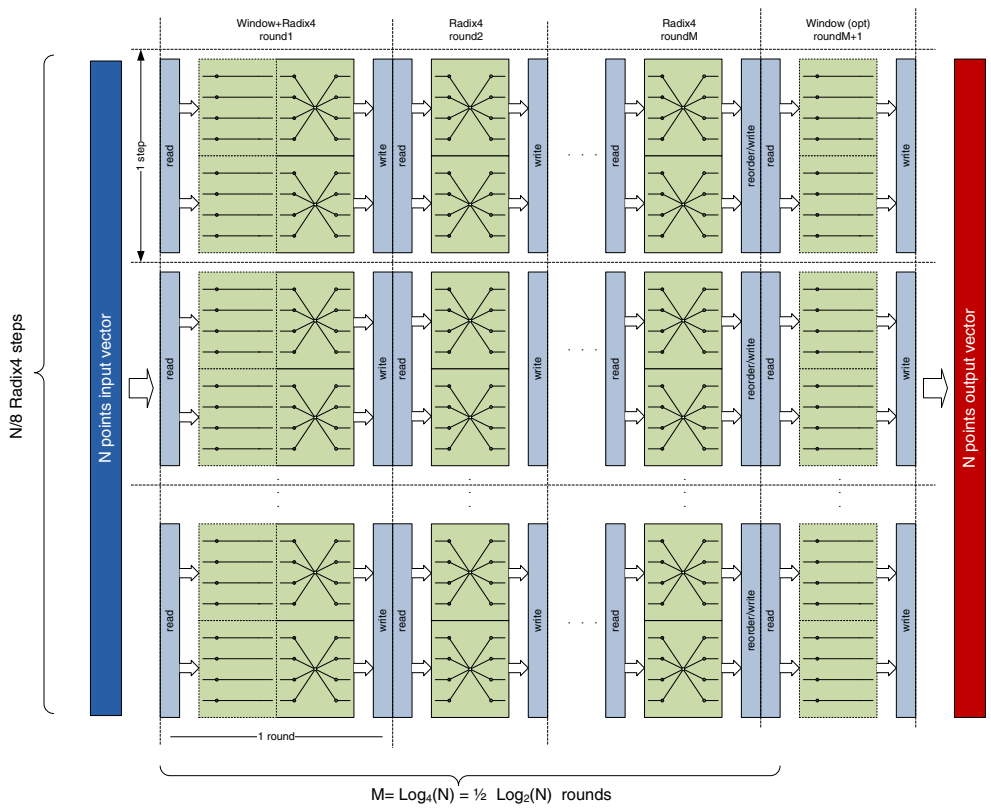


Figure 45-29. Principle of Operation

The initial round can be combined with window operation, and this extra window operation will not take additional cycles. There is a WIN bit in the FFT instruction which can be used in the first stage as all FFT twiddles are 1. Another window round can be appended after the final Radix round, which requires additional steps.

Table 45-34. FFT stages

FFT Size	Steps	Radix4 Rounds	Radix2 Rounds(last round)	Radix kernel invocations
8	1	1	1	2
16	2	2	0	4
32	4	2	1	12
64	8	3	0	24
128	16	3	1	64
256	32	4	0	128
512	64	4	1	320

Table continues on the next page...

Table 45-34. FFT stages (continued)

FFT Size	Steps	Radix4 Rounds	Radix2 Rounds(last round)	Radix kernel invocations
1024	128	5	0	640
2048	256	5	1	1536
4096	512	6	0	3072

45.9.7.7 Window Multiplication

The window operation uses the same Radix4 kernel, but bypasses the butterfly after complex multiplications. The window operation can be done separately using a WIN command or as a part of RDX4 command. When window operation is done as a part of RDX4 command it is allowed only in the first round. To enable a complete complex vector multiplication, the first twiddle factor can be chosen different from 1.

There are three types of vector multiplications:

- with a single constant
- with a full vector, of the same size as the operand vector
- with a partial vector of a smaller size, and repeated elements

The constant for multiplication can be located in twiddle RAM or work register or immediate value (as part of the command). For a full vector or partial vector multiplication the coefficients are stored in the twiddle RAM. The partial vector multiplication uses an address offset and a modulo to get each element of the multiplier vector. The full vector multiplication multiplies each element from the operand vector with an element of multiplier vector. The address of the elements of the multiplier vector is incrementing.

Table 45-35 shows the different memory address calculation modes for window multiplication.

Table 45-35. Memory address calculation modes for window muliplication

Mode	Action
0	multiply with a single 32-bit complex immediate value
1	multiply with a single 32-bit complex constant from coefficient address (twiddle RAM address or work register index).
2	increment and modulus applied after every Radix4 step. coeff_addr = coeff_base_addr+ ((step*MCA_INC) % MCA_MOD) * 8 + m where step denotes the step number. This step number = 0 in the first clock cycle, 1 in the second clock cycle, etc.

Table continues on the next page...

Table 45-35. Memory address calculation modes for window muliplication (continued)

	MCA_INC = the address increment value MCA_MOD = modulo value m = 0 to 7, is the index of the address generated in a single step (out of the 8 addresses)
3	Reserved

For a full vector multiplication the window multiplication mode should be =2 and MCA_MOD = 0.

When window operation is done as a part of RDX4 command the window coefficients need to be stored in a continuous, reordered manner. This pattern is described for various vector lengths in the figures shown below. In this particular case the window multiplication mode should be = 2, MCA_INC = 1 and MCA_MOD = 0. However an exception is the combined FFT16 operation where if the same window is needed to be used for all the different sets of operands, the MCA_MOD field should be set to 1. Similarly for combined FFT32 operation where also the same window is needed to be used for all the different sets of operands, and so the MCA_MOD field should be set to 2. CA stands for Column address in the figures below.

FFT 8

Columns	CA0
Slice 0	0
Slice 1	2
Slice 2	4
Slice 3	6
Slice 4	1
Slice 5	3
Slice 6	5
Slice 7	7

Figure 45-30. Window Coefficients in Twiddle RAM for 8 point FFT

FFT 16

Columns	CA0	CA1
Slice 0	0	2
Slice 1	4	6
Slice 2	8	10
Slice 3	12	14
Slice 4	1	3
Slice 5	5	7
Slice 6	9	11
Slice 7	13	15

FFT 32

Columns	CA0	CA1	CA2	CA3
Slice 0	0	4	1	5
Slice 1	8	12	9	13
Slice 2	16	20	17	21
Slice 3	24	28	25	29
Slice 4	2	6	3	7
Slice 5	10	14	11	15
Slice 6	18	22	19	23
Slice 7	26	30	27	31

FFT 64

Columns	CA0	CA1	CA2	CA3	CA4	CA5	CA6	CA7
Slice 0	0	1	2	3	8	9	10	11
Slice 1	16	17	18	19	24	25	26	27
Slice 2	32	33	34	35	40	41	42	43
Slice 3	48	49	50	51	56	57	58	59
Slice 4	4	5	6	7	12	13	14	15
Slice 5	20	21	22	23	28	29	30	31
Slice 6	36	37	38	39	44	45	46	47
Slice 7	52	53	54	55	60	61	62	63

Figure 45-31. Window Coefficients in Twiddle RAM for 16, 32 and 64 point FFT

FFT 128

Columns	CA0	-	CA7	CA8	-	CA15
Slice 0	0	-	7	16	-	23
Slice 1	32	-	39	48	-	55
Slice 2	64	-	71	80	-	87
Slice 3	96	-	103	112	-	119
Slice 4	8	-	15	24	-	31
Slice 5	40	-	47	56	-	63
Slice 6	72	-	79	88	-	95
Slice 7	104	-	111	120	-	127

FFT 256

Columns	CA0	-	CA15	CA16	-	CA31
Slice 0	0	-	15	32	-	47
Slice 1	64	-	79	96	-	111
Slice 2	128	-	143	160	-	175
Slice 3	192	-	207	224	-	239
Slice 4	16	-	31	48	-	63
Slice 5	80	-	95	112	-	127
Slice 6	144	-	159	176	-	191
Slice 7	208	-	223	240	-	255

FFT 512

Columns	CA0	-	CA31	CA32	-	CA63
Slice 0	0	-	31	64	-	95
Slice 1	128	-	159	192	-	223
Slice 2	256	-	287	320	-	351
Slice 3	384	-	415	448	-	479
Slice 4	32	-	63	96	-	127
Slice 5	160	-	191	224	-	255
Slice 6	288	-	319	352	-	383
Slice 7	416	-	447	480	-	511

FFT 1024

Columns	CA0	-	CA63	CA64	-	CA127
Slice 0	0	-	63	128	-	191
Slice 1	256	-	319	384	-	447
Slice 2	512	-	575	640	-	703
Slice 3	768	-	831	896	-	959
Slice 4	64	-	127	192	-	255
Slice 5	320	-	383	448	-	511
Slice 6	576	-	639	704	-	767
Slice 7	832	-	895	960	-	1023

FFT 2048

Columns	CA0	-	CA127	CA128	-	CA255
Slice 0	0	-	127	256	-	383
Slice 1	512	-	639	768	-	895
Slice 2	1024	-	1151	1280	-	1407
Slice 3	1536	-	1663	1792	-	1919
Slice 4	128	-	255	384	-	511
Slice 5	640	-	767	896	-	1023
Slice 6	1152	-	1279	1408	-	1535
Slice 7	1664	-	1791	1920	-	2047

FFT 4096

Columns	CA0	-	CA255	CA256	-	CA511
Slice 0	0	-	255	512	-	767
Slice 1	1024	-	1279	1536	-	1791
Slice 2	2048	-	2303	2560	-	2815
Slice 3	3072	-	3327	3584	-	3839
Slice 4	256	-	511	768	-	1023
Slice 5	1280	-	1535	1792	-	2047
Slice 6	2304	-	2559	2816	-	3071
Slice 7	3328	-	3583	3840	-	4095

Figure 45-32. Window Coefficients in Twiddle RAM for 128, 256, 512, 1024, 2048 and 4096 point FFT

For the WIN command of the RDX4 or IRDX4 where the windowing option is enabled in round=0, it is possible to specify the type of window coefficients as complex, real only or imaginary only. In case of real only window it is assumed that the coefficients are all real and stored only in the real part of the TRAM memory locations. In case of imaginary only window it is assumed that the coefficients are all real and stored only in the imaginary part of the TRAM memory locations.

45.9.7.8 FIR Filter

The command word for FIR filter must give the twiddle RAM address where the tap coefficients are stored. These taps are stored in all the slices of the twiddle RAM at that particular address. If the number of taps is less than eight then the remaining taps are assumed to be zero. The tap stored at slice0 of the twiddle RAM is the tap which gets the undelayed input from the operand RAM.

For the FIR command it is possible to specify the type of tap coefficients as complex, real only or imaginary only. In case of real only taps it is assumed that the coefficients are all real and stored only in the real part of the TRAM memory locations. In case of imaginary only taps it is assumed that the coefficients are all real and stored only in the imaginary part of the TRAM memory locations.

It is intended that the tap coefficients or operands be filled such that they give unity gain for filter output. This gain is defined with respect to a 4-tap filter. Thus if an 8-tap filter is being used then the tap coefficients can be half the magnitude. If a 2-tap filter is used then to achieve unity gain the tap coefficients can be similar to the 4-tap case but the operands are doubled by enforcing a left-shift by 1 position through the command word.

There are two types of intializations:

- Zero run-in initialization where the unoccupied locations contain zeroes.
- Constant run-in initialization where the unoccupied locations contain the value of the first input operand.

45.9.7.9 FFT Special Usage

45.9.7.9.1 Inverse FFT

The FFT kernel also supports the inverse radix4 and inverse radix2 operations through the IRDX4 and IRDX2 commands.

The FFT kernel calculates the inverse radix according to the following equation:

$$f(n) = \sum_{k=0}^{N-1} F(k)\, e^{+j2\pi nk/N}$$

Figure 45-33. Inverse radix equation

The inverse radix commands are similar to the RDX4 and RDX2 commands, however RDX2 command supports the REAL_FFT bit, but there is no such corresponding bit in IRDX2.

45.9.7.9.2 Combined FFT8, FFT16 and FFT32

With a low round and step count, the pipeline stages limit the reachable performance. Fewer pipeline stages would result in an increase in hardware and power. This drawback appears especially with a high count of FFT8, FFT16 and FFT32 operations.

To overcome this issue, FFT8, FFT16 or FFT32 operations can be aggregated. The operations are combined in a way that the first round is executed together for a number of different sets of input data, as defined by the vector length. The vector length must be a value dividable by 8,16 or 32, as the case may be. After the complete vector has been processed in the first round, the second round and third round(for FFT32) are executed. The result of this operation gives N times the individual FFTs, where N which is the vector length divided by 8,16 or 32, also denotes the number of aggregated sets of input data. The speed-up results from the fact that pipelining contributes only once to the operation time within a round.

Note: Even when the combined FFT8, FFT16 or FFT32 option is in use, only 8, 16 or 32 twiddles respectively need to be stored in all the twiddle RAM slices regardless of the vector size.

45.9.7.9.3 2^odd FFT

FFTs with a size that is an odd power of two (e.g. 8, 32, 128, 512, 2048) cannot be built from Radix4 operations solely. They are constructed from Radix4 rounds and a final Radix2 round. For instance, 512 point FFT can be constructed by 4 RDX4 instructions(0-3 rounds) and last round(4) as RDX2 instruction.

45.9.7.9.4 Real Number FFT

A lot of typical FFT operations in RADAR processing are just with real input data. The FFT algorithm is defined in a generic way and does not need to be aware of input data with all zero imaginary components, however a real only FFT can be about twice as fast than its complex counterpart. There are two basic options to implement a real number only N-point FFT:

- by extending the real number with a zero imaginary part and performing the N point FFT
- by mixing two N point real data to N point complex data and performing N point complex FFT. Then split the 2 outputs by using Radix2 command with REAL_FFT field set.

The first option has trivial implementation, however it suffers performance bottlenecks. The second option generates a complex vector by setting the real part to vector1 and the imaginary part to vector2. Afterwards, an complex N point FFT is performed, which delivers a merged result for the FFTs of both input vectors. A final decomposition round is required to separate the results for both input vectors. Depending on the vector size, the second option is nearly twice as efficient as the first option, only the decomposition round represents a small penalty.

The first option is the default option, when real data are used. The complex part is set to zero during DMA transfer or by an explicit copy instruction. The second option requires an additional copy instruction to construct the input vector and an additional round with Radix2 butterflies(REAL_FFT set).

45.9.7.9.5 Adaptive scaling

The adaptive scaling feature is optional for rdx2 and rdx4 and also irdx4 and irdx2 commands, and is not applicable to FIR and WIN and SCP commands.

For rdx2 and rdx4 and also irdx4 and irdx2 operations, as output data is being written to the OPRAM, an MSB occupation counter checks the number of unoccupied MSB bits in the entire output data chunk. This counter value gives an idea of the attenuation of the output and therefore also of the amount of magnification possible.

In the next round the input data before being given to the butterflies is shifted left by the amount previously calculated by the MSB occupation counter. This helps to increase the resolution of the FFT operation.

During adaptive scaling, in round=0 only, the normal pre-shift value specified by the command word is given to the barrel shifter before the butterflies. In round>0 the value calculated by the MSB occupation counter is given to the barrel shifter, the pre-shift value is ignored.

The value of MSB occupation counter is not allowed to exceed 4 in any cycle. Thus the total shift cannot exceed 24 left shifts.

At the end of every command the accumulated value of shifts that have already occurred is stored in the work register specified in the command word. Also, each work register can store 8 such accumulated values at different offsets. This offset value is also supplied in the command word. The next round shift value remains stored only in the MSB occupation counter. This makes it necessary that the different rounds of the FFT operation be performed continuously one after the other.

The command word also decides whether the MSB occupation counter counts unoccupied bits from bit 23 downwards or from bit 15 downwards.

It should be noted that in case of indirect memory access, the work register chosen to read source and destination addresses should not be the same as the register to which the accumulated shift values are stored at the end of each command

Figure 45-34 shows the FFT operation with adaptive scaling enabled.

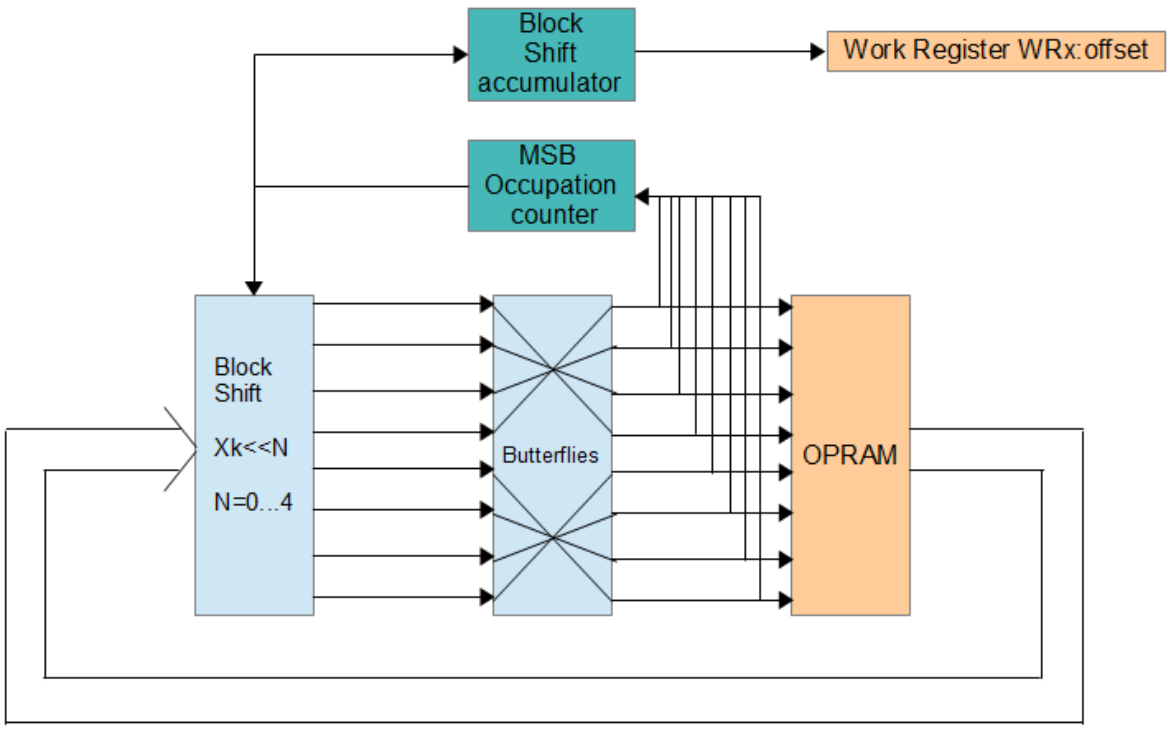


Figure 45-34. FFT operation with adaptive scaling enabled

45.9.7.9.6 Scalar Product

Scalar Product operation can be performed using the Radix 4 or Radix 2 kernels by executing the SCP command. Scalar product is basically a multiply-add operation. SCP allows scalar product operations with 2, 4 or 8 coefficients. The command requires eight operands each cycle, so the vector length should be a multiple of eight. When 2-coefficient scalar product is calculated then four output values are given out each cycle and stored at consecutive locations. When 4-coefficient scalar product is calculated then two output values are given out each cycle and stored at consecutive locations. When 8-coefficient scalar product is calculated then one output values is given out each cycle and stored at consecutive locations.

The 8-coefficient operation is possible by a slight modification in the Radix4 butterfly structure. The complex multipliers multiply the coefficients with the inputs operands and the eight products from the two butterflies are then added together.

The command word for SCP must give the twiddle RAM address where the coefficients are stored. In case of 4-coefficient case the coefficients come from slice0 to slice3 of TRAM for both the products. In case of 2-coefficient case the coefficients come from slice0 and slice1 of TRAM for all the four products.

For the SCP command it is possible to specify the type of TRAM coefficients as complex, real only or imaginary only. In case of real only coefficients it is assumed that the coefficients are all real and stored only in the real part of the TRAM memory locations. In case of imaginary only coefficients it is assumed that the coefficients are all real and stored only in the imaginary part of the TRAM memory locations.

It is intended that the coefficients or operands be filled such that they give unity gain for filter output. This gain is defined with respect to a 4-coefficient scalar product. Thus if an 8-coefficient operation is being done then the coefficients can be half the magnitude. If a 2-coefficient operation is done then to achieve unity gain the coefficients can be similar to the 4-coefficient case but the operands are doubled by enforcing a left-shift by 1 position through the command word.

Figure 45-35 shows the kernel operation for 8-coefficient, 4-coefficient and 2-coefficient operations.

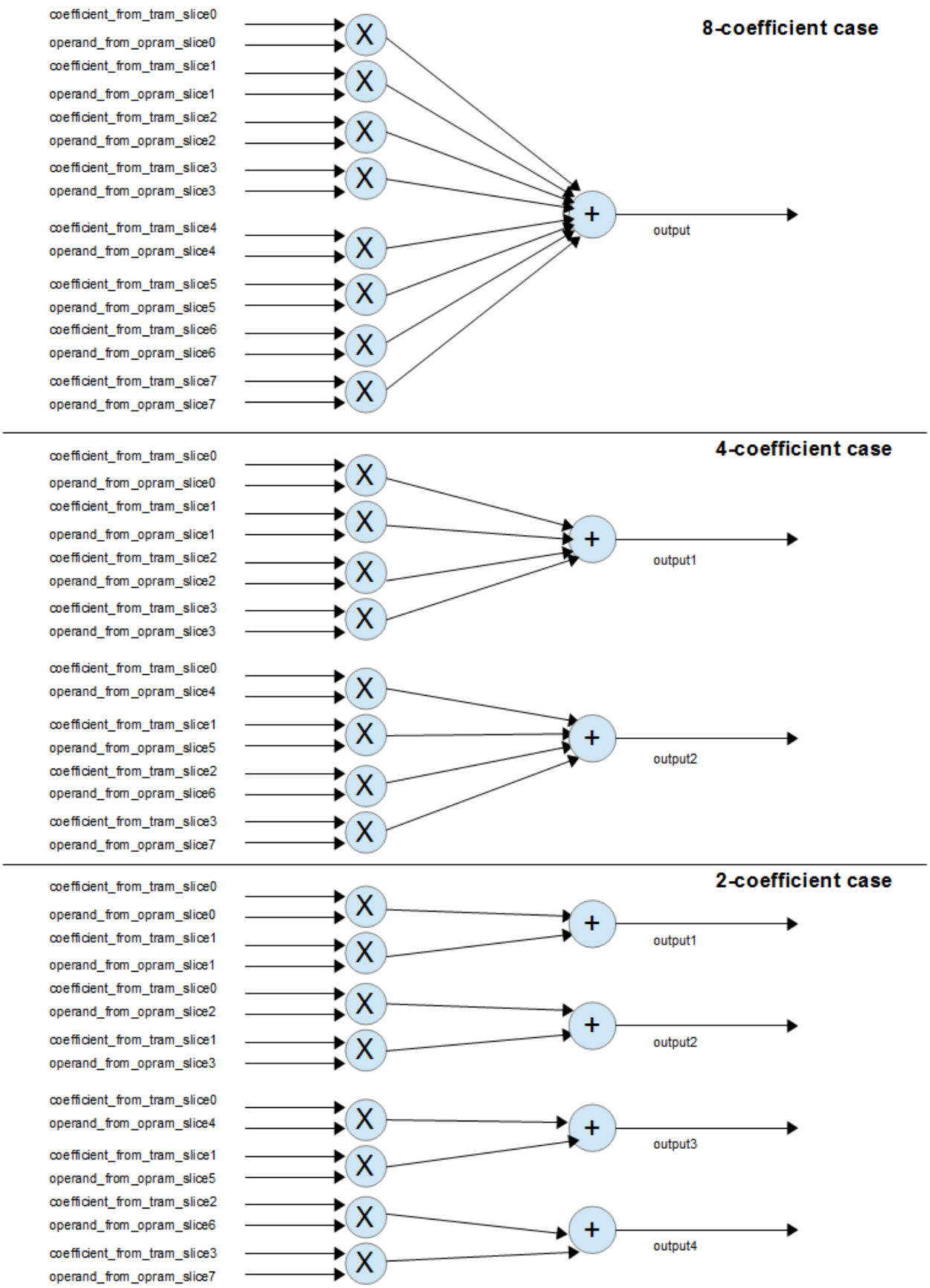


Figure 45-35. Kernel operations for 8-coefficient, 4-coefficient and 2-coefficient SCP

operations

Figure 45-36 shows the cycle-by-cycle SCP operation for the case NO_OF_COEFFICIENTS=8, VEC_SZ=24.

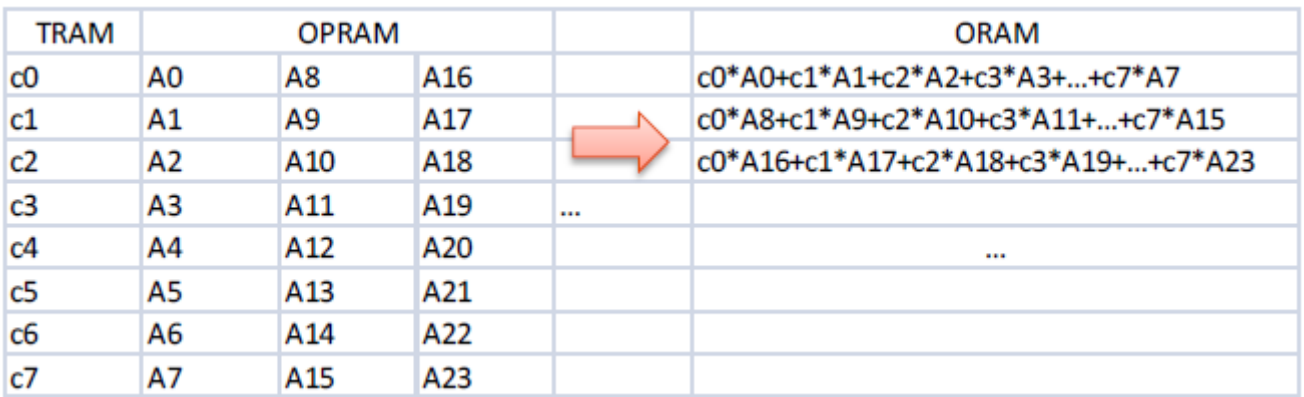


Figure 45-36. SCP operation for 8-coefficient and vector length=24 case

45.9.7.10 FFT Instructions Format

WIN Instruction

The table below shows the instruction bit encoding used for Window(WIN) Command. The details of each bit field and their operation is defined in the sections below.

Table 45-36. WIN Instruction (MULT_MODE !=00)

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100001)						IN_DATTYP	WIN_TYPE		Reserved						
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCA_MOD				SHFTVAL_OFST			MCA_INC								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

Table 45-36. WIN Instruction (MULT_MODE !=00) (continued)

SHFTVAL_WR_NO	MULT_MOD	SHFT VAL_ SRC	SHFT_VAL	Reserved

Table 45-37. WIN Instruction bit description

Bits	Bit Field	Description
[127:122]	OPCODE	100001 - WIN
[121]	IN_DATTYP	Input data type
		0 - Real
		1 - Complex
[120:119]	WIN_TYPE	Indicates the type of window coefficients
		00 - complex window coefficients
		01 - complex window coefficients
		10 - window is real and stored in the imaginary part of every TRAM memory location
		11 - window is real and stored in the real part of every TRAM memory location
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	MULT_COEF_ADD	Twiddle or Multiplier Coefficient address
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[31:28]	MCA_MOD	Multiply Coefficient Address modulo value, modulus is 2^n this entry (only valid with window)
		0000 - no modulo calculation
		0001- modulo 2
		0010- modulo 4
		0011 - modulo 8
		0100 - modulo 16
		0101 - modulo 32
		0110 - modulo 64
		0111 - modulo 128
		1000 - modulo 256
		1001 to 1111 - modulo 512
[27:25]	SHFTVAL_OFST	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained

Table continues on the next page...

Table 45-37. WIN Instruction bit description (continued)

Bits	Bit Field	Description
		000 - Value comes from bits 5:0 of WR
		001 - Value comes from bits 11:6 of WR
		010 - Value comes from bits 17:12 of WR
		011 - Value comes from bits 23:18 of WR
		100 - Value comes from bits 29:24 of WR
		101 - Value comes from bits 35:30 of WR
		110 - Value comes from bits 41:36 of WR
		Value comes from bits 47:42 of WR111 -
[24:16]	MCA_INC	Multiply Coefficient Address Increment, Address offset from coefficient to coefficient
[15:10]	SHFTVAL_WR_NO	Gives the index of Work Register from where the pre-scaling left shift value is obtained
[9:8]	MULT_MOD	Mult Mode
		00 - Mode 0 - multiply with a single 32-bit complex immediate value
		01- Mode 1 - multiply with a single 32-bit complex constant from coefficient address (twiddle RAM address or work register index).
		10- Mode 2 - increment and modulus applied after every Radix4 step. coeff_addr = coeff_base_addr + ((step*MCA_INC) % MCA_MOD) * 8 + m where step denotes the step number. This step number = 0 in the first clock cycle, 1 in the second clock cycle, etc. MCA_INC = the address increment value MCA_MOD = modulo value m = 0 to 7, is the index of the address generated in a single step (out of the 8 addresses)
		11- Reserved
[7]	SHFTVAL_SRC	Gives the source from where the pre-scaling left shift value is obtained. This bit should be SET only if the MULT_MOD field !=0, else there is an error indication
		0 - Value comes from the command word
		1 - Value comes from the Work Register
[6:4]	SHFT_VAL	Pre-scaling left shift value
		000 - No shift
		001 - Left shift 1 bit
		010 - Left shift 2 bits
		011 - Left shift 3 bits
		100 - Left shift 4 bits
		101 to 111 - Left shift 8 bits

The table below shows the instruction bit encoding used for Window(WIN) command when MULT_MODE = 0. Only CC_IM and CC_RE field differs from the last instruction format shown above.

Table 45-38. WIN Instruction (MULT_MODE =00)

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
CC_IM[15:0]															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC_RE[15:0]															

Table 45-39. WIN Instruction (MULT_MODE =00) Instruction bit description

Bits	Bit Field	Description
[63:48]	CC_IM[15:0]	Constant Mult Coefficient Imaginary part
[31:16]	CC_RE[15:0]	Constant Mult Coefficient Real part

Radix4 Instruction

The table below shows the instruction bit encoding used for Radix4(RDX4) Command. The details of each bit field are defined in the sections below. For MULT_MODE=0, refer to fields will be as described in WIN Instruction (MULT_MODE =00)

Table 45-40. RDX4 Instruction (MULT_MODE !=00)

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100010)						IN_DATTYP	WIN_TYPE		FFT_RND			Reserved	ADPTV	ADPTV_SHFT	QUAD_EXT
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
REPEAT		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table 45-40. RDX4 Instruction (MULT_MODE !=00) (continued)

MCA_MOD				ADPTV_OFFSET			MCA_INC								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADPTV_WRNUM						MULT_MOD	SHFT VAL_ SRC	SHFT_VAL				TW_OVS			

Table 45-41. RDX4 Instruction bit description

Bits	Bit Field	Description
[121]	IN_DATTYP	Input data type
		0 - Real
		1 - Complex
[120:119]	WIN_TYPE	Indicates the type of window coefficients
		00 - Windowing is not enabled
		01 - complex window coefficients
		10 - window is real and stored in the imaginary part of every TRAM memory location
		11 - window is real and stored in the real part of every TRAM memory location
[118:116]	FFT_RND	FFT Round
		000-Round 0 (Reversed Input Stage)
		001-Round 1 (Highest for FFT16 & 32)
		010-Round 2 (Highest for FFT64 & 128)
		011-Round 3 (Highest for FFT256 & FFT512)
		100-Round 4 (Highest for FFT1024 & FFT 2046)
		101-Round 5 (Highest for FFT4096)
		110 and 111-Reserved
[114]	ADPTV	0 - Adaptive scaling not enabled
		1 - Adaptive scaling enabled
[113]	ADPTV_SHFT	1 - MSB occupation counter counts unoccupied bits from bit 23 downwards
		0 - MSB occupation counter counts unoccupied bits from bit 15 downwards
[112]	QUAD_EXT	Quadrature Extension
		0 – No quadrature extension
		1 – Quadrature extension used
[111:110]	REPEAT	Decides whether combined fft operation is enabled or not
		00 - No combined fft operation
		01 - Combined FFT8, for vector length/8 x FFTs
		10 - Combined FFT16, for vector length/16 x FFTs
		11 - Combined FFT32, for vector length/32 x FFTs
[109]	IMA	Indirect memory addressing

Table continues on the next page...

Table 45-41. RDX4 Instruction bit description (continued)

Bits	Bit Field	Description
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	MULT_COEF_ADD	Twiddle or Multiplier Coefficient address
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[31:28]	MCA_MOD	Multiply Coefficient Address modulo value, modulus is 2^n this entry (only valid with window)
		0000 - no modulo calculation
		0001- modulo 2
		0010- modulo 4
		0011- modulo 8
		0100- modulo 16
		0101- modulo 32
		0110- modulo 64
		0111- modulo 128
		1000- modulo 256
		1001 to 1111- modulo 512
[27:25]	ADPTV_OFFSET when SHFTVAL_SRC = 0 & FFT_RND > 0	This offset value gives the location in a Work Register where the accumulated Block Shift value is written at the end of every fft round
		000 - Value written in bits 5:0 of Work Register
		001 - Value written in bits 11:6 of Work Register
		010 - Value written in bits 17:12 of Work Register
		011 - Value written in bits 23:18 of Work Register
		100 - Value written in bits 29:24 of Work Register
		101 - Value written in bits 35:30 of Work Register
		110 - Value written in bits 41:36 of Work Register
		111 - Value written in bits 47:42 of Work Register
	ADPTV_OFFSET when SHFTVAL_SRC = 1 & FFT_RND = 0	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained
		000 - Value comes from bits 5:0 of WR
		001 - Value comes from bits 11:6 of WR
		010 - Value comes from bits 17:12 of WR
		011 - Value comes from bits 23:18 of WR
		100 - Value comes from bits 29:24 of WR
		101 - Value comes from bits 35:30 of WR
		110 - Value comes from bits 41:36 of WR

Table continues on the next page...

Table 45-41. RDX4 Instruction bit description (continued)

Bits	Bit Field	Description
		111 - Value comes from bits 47:42 of WR
[24:16]	MCA_INC	Multiply Coefficient Address Increment, Address offset from coefficient to coefficient
[15:10]	ADPTV_WRNUM when SHFTVAL_SRC = 0 & FFT_RND > 0	Gives the Work Register index where the accumulated Block Shift value is written at the end of every fft round
	ADPTV_WRNUM when SHFTVAL_SRC = 1 & FFT_RND = 0	Gives the index of Work Register from where the pre-scaling left shift value is obtained
[9:8]	MULT_MOD	Mult Mode
		00 - Mode 0 - multiply with a single 32-bit complex immediate value
		01- Mode 1 - multiply with a single 32-bit complex constant from coefficient address (twiddle RAM address or work register index).
		10- Mode 2 - increment and modulus applied after every Radix4 step. coeff_addr = coeff_base_addr + ((step*MCA_INC) % MCA_MOD) * 8 + m where step denotes the step number. This step number = 0 in the first clock cycle, 1 in the second clock cycle, etc. MCA_INC = the address increment value MCA_MOD = modulo value m = 0 to 7, is the index of the address generated in a single step (out of the 8 addresses)
		11- Reserved
[7]	SHFTVAL_SRC	Gives the source from where the pre-scaling left shift value is obtained. In case WIN_TYPE > 0 this field should be SET only if MULT_MOD != 0, else there is an error indication.
		0 - Value comes from the command word
		1 - Value comes from Work Register
[6:4]	SHFT_VAL	Pre-scaling left shift value
		000 – no shift
		001 – shift left 1 bit
		010 - shift left 2 bits
		011 - shift left 3 bits
		100 - shift left 4 bits
		101 to 111 - shift left 8 bits
[2:0]	TW_OVS	Twiddle Oversampling - to use stored Wn with n > k*vector length
		0000 – no oversampling
		0001 – oversampling factor of 2
		0010 – oversampling factor of 4
		0011 – oversampling factor of 8
		0100 – oversampling factor of 16
		0101 - oversampling factor of 32

Table continues on the next page...

Table 45-41. RDX4 Instruction bit description (continued)

Bits	Bit Field	Description
		0110 – oversampling factor of 64
		0111 – oversampling factor of 128
		1000 – oversampling factor of 256
		1001 to 1111 - oversampling factor of 512

Radix2 Instruction

The table below shows the instruction bit encoding used for Radix2(RDX2) Command. The details of each bit field is defined in the sections below.

Table 45-42. RDX2 Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100011)						IN_DATTYP	Reserved		FFT_RND			REAL_FFT	ADPTV	ADPTV_SHFT	QUAD_EXT
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
REPEAT		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ADPTV_OFFSET			Reserved								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADPTV_WRNUM						Reserved		SHFTVAL_SRC	SHFT_VAL			TW_OVS			

Table 45-43. RDX2 Instruction bit description

Bits	Bit Field	Description
[121]	IN_DATTYP	Input data type
		0 - Real
		1 - Complex
[118:116]	FFT_RND	FFT Round - Last rounds only as mentioned

Table continues on the next page...

Table 45-43. RDX2 Instruction bit description (continued)

Bits	Bit Field	Description
		000 - Reserved
		001 - Round 1 (for 8 point FFT)
		010- Round 2 (for 32 point FFT)
		011- Round 3 (for 128 point FFT)
		100- Round 4 (for 512 point FFT)
		101- Round 5 (for 2048 point FFT)
		110 and 111 - Reserved
[115]	REAL_FFT	When set, RDX2 command is used for splitting Operands corresponding to 2 Real input vectors.
		0 - Normal Radix2 Command
		1- Radix2 command used for operands split(Twiddles used will be 1 and fields TW_OVS & QUAD_EXT will be ignored)
[114]	ADPTV	0 - Adaptive scaling not enabled
		1 - Adaptive scaling enabled
[113]	ADPTV_SHFT	0 - MSB occupation counter counts unoccupied bits from bit 15 downwards
		1 - MSB occupation counter counts unoccupied bits from bit 23 downwards
[112]	QUAD_EXT	Quadrature Extension
		0 – No quadrature extension
		1 – Quadrature extension used
[111:110]	REPEAT	Decides whether combined fft operation is enabled or not
		00 - No combined fft operation
		01 - Combined FFT8, for vector length/8 x FFTs
		10 - Reserved
		11 - Combined FFT32, for vector length/32 x FFTs
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	MULTI_COEF_ADD	Twiddle or Multiplier Coefficient address
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[27:25]	ADPTV_OFFSET when SHFTVAL_SRC = 0 & FFT_RND > 0	This offset value gives the location in a Work Register where the accumulated Block Shift value is written at the end of every fft round
		000 - Value written in bits 5:0 of Work Register
		001 - Value written in bits 11:6 of Work Register

Table continues on the next page...

Table 45-43. RDX2 Instruction bit description (continued)

Bits	Bit Field	Description
		010 - Value written in bits 17:12 of Work Register
		011 - Value written in bits 23:18 of Work Register
		100 - Value written in bits 29:24 of Work Register
		101 - Value written in bits 35:30 of Work Register
		110 - Value written in bits 41:36 of Work Register
		111 - Value written in bits 47:42 of Work Register
[15:10]	ADPTV_WRNUM when SHFTVAL_SRC = 0 & FFT_RND > 0	Gives the Work Register index where the accumulated Block Shift value is written at the end of every fft round
[7]	SHFTVAL_OFST	Gives the source from where the pre-scaling left shift value is obtained
		0 - Value comes from the command word
		1 - Value comes from Work Register
[6:4]	SHFT_VAL	Pre-scaling left shift value
		000 – no shift
		001 – shift left 1 bit
		010 - shift left 2 bits
		011 - shift left 3 bits
		100 - shift left 4 bits
		101 to 111 - shift left 8 bits
[3:0]	TW_OVS	Twiddle Oversampling - to use stored Wn with n > k*vector length
		0000 – no oversampling
		0001 – oversampling factor of 2
		0010 – oversampling factor of 4
		0011 - oversampling factor of 8
		0100 - oversampling factor of 16
		0101 - oversampling factor of 32
		0110 - oversampling factor of 64
		0111 - oversampling factor of 128
		1000 - oversampling factor of 256
		1001 to 1111 - oversampling factor of 512

FIR Instruction

The table below shows the instruction bit encoding used for FIR Command. The details of each bit field is defined in the sections below.

Table 45-44. FIR Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112

Table continues on the next page...

Table 45-44. FIR Instruction (continued)

OPCODE(101001)						IN_DAT_TYP	WIN_TYPE		Reserved		INIT	NO_OF_TAPS			
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
TAP_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				SHFTVAL_OFST		Reserved									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHFTVAL_WR_NO						Reserved		SHFTVAL_SRC	SHFT_VAL			Reserved			

Table 45-45. FIR Instruction bit description

Bits	Bit Field	Description
[121]	IN_DAT_TYP	Input data type
		0- Real data
		1 - Complex data
[120:119]	WIN_TYPE	Indicates the type of window coefficients
		00 - complex window coefficients
		01 - complex window coefficients
		10 - window is real and stored in the imaginary part of every TRAM memory location
		11 - window is real and stored in the real part of every TRAM memory location
[116]	INIT	Decides which type of initialization process is to be performed during FIR operation
		0 - Zero run-in initialization performed
		1 - Constant run-in initialization performed
[115:112]	NO_OF_TAPS	Gives the number of taps in the FIR filter
		0000 - 2 tap filter
		0001 - 2 tap filter

Table continues on the next page...

Table 45-45. FIR Instruction bit description (continued)

Bits	Bit Field	Description
		0010 - 2 tap filter
		0011 - 3 tap filter
		0100 - 4 tap filter
		0101 - 5 tap filter
		0110 - 6 tap filter
		0111 - 7 tap filter
		1000 to 1111 - 8 tap filter
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:98]	VEC_SZ	Vector size/length, should be a multiple of 8
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	TAP_COEFF_ADD	Gives the twiddle RAM address where the tap coefficients for the FIR filter are stored
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[27:25]	SHFTVAL_OFST	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained
		000 - Value comes from bits 5:0 of WR
		001 - Value comes from bits 11:6 of WR
		010 - Value comes from bits 17:12 of WR
		011 - Value comes from bits 23:18 of WR
		100 - Value comes from bits 29:24 of WR
		101 - Value comes from bits 35:30 of WR
		110 - Value comes from bits 41:36 of WR
		111 - Value comes from bits 47:42 of WR
[15:10]	SHFTVAL_WR_NO	Gives the index of Work Register from where the pre-scaling left shift value is obtained
[7]	SHFTVAL_SRC	Gives the source from where the pre-scaling left shift value is obtained
		0 - Value comes from the command word

Table continues on the next page...

Table 45-45. FIR Instruction bit description (continued)

Bits	Bit Field	Description
		1 - Value comes from Work Register
[6:4]	SHFT_VAL	Pre-scaling before multiplication
		000 - No shift
		001 - Left shift by 1 bit
		010 - Left shift by 2 bits
		011 - Left shift by 3 bits
		100 - Left shift by 4 bits
		101 to 111 - Left shift by 8 bits

45.9.7.11

Inverse Radix4 Instruction

The table below shows the instruction bit encoding used for Inverse Radix4(IRDX4) Command. The details of each bit field are defined in the sections below. For MULT_MODE=0, refer to fields will be as described in WIN Instruction (MULT_MODE =00)

Table 45-46. IRDX4 Instruction (MULT_MODE !=00)

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(101011)						IN_DATTYP	WIN_TYPE		FFT_RND			Reserved	ADPTV	ADPTV_SHFT	QUAD_EXT
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
REPEAT		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCA_MOD				ADPTV_OFFSET				MCA_INC							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADPTV_WRNUM						MULT_MOD		SHFTVAL_SRC	SHFT_VAL			TW_OVS			

Table 45-47. IRDX4 Instruction bit description

Bits	Bit Field	Description
[121]	IN_DATTYP	Input data type
		0 - Real
		1 - Complex
[120:119]	WIN_TYPE	Indicates the type of window coefficients
		00 - Windowing is not enabled
		01 - complex window coefficients
		10 - window is real and stored in the imaginary part of every TRAM memory location
		11 - window is real and stored in the real part of every TRAM memory location
[118:116]	FFT_RND	FFT Round
		000-Round 0 (Reversed Input Stage)
		001-Round 1 (Highest for FFT16 & 32)
		010-Round 2 (Highest for FFT64 & 128)
		011-Round 3 (Highest for FFT256 & FFT512)
		100-Round 4 (Highest for FFT1024 & FFT 2046)
		101-Round 5 (Highest for FFT4096)
		110 and 111-Reserved
[114]	ADPTV	0 - Adaptive scaling not enabled
		1 - Adaptive scaling enabled
[113]	ADPTV_SHFT	1 - MSB occupation counter counts unoccupied bits from bit 23 downwards
		0 - MSB occupation counter counts unoccupied bits from bit 15 downwards
[112]	QUAD_EXT	Quadrature Extension
		0 – No quadrature extension
		1 – Quadrature extension used
[111:110]	REPEAT	Decides whether combined fft operation is enabled or not
		00 - No combined fft operation
		01 - Combined FFT8, for vector length/8 x FFTs
		10 - Combined FFT16, for vector length/16 x FFTs
		11 - Combined FFT32, for vector length/32 x FFTs
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	MULT_COEF_ADD	Twiddle or Multiplier Coefficient address

Table continues on the next page...

Table 45-47. IRDX4 Instruction bit description (continued)

Bits	Bit Field	Description
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[31:28]	MCA_MOD	Multiply Coefficient Address modulo value, modulus is 2^n this entry (only valid with window)
		0000 - no modulo calculation
		0001- modulo 2
		0010- modulo 4
		0011- modulo 8
		0100- modulo 16
		0101- modulo 32
		0110- modulo 64
		0111- modulo 128
		1000- modulo 256
		1001 to 1111- modulo 512
[27:25]	ADPTV_OFFSET when SHFTVAL_SRC = 0 & FFT_RND > 0	This offset value gives the location in a Work Register where the accumulated Block Shift value is written at the end of every fft round
		000 - Value written in bits 5:0 of Work Register
		001 - Value written in bits 11:6 of Work Register
		010 - Value written in bits 17:12 of Work Register
		011 - Value written in bits 23:18 of Work Register
		100 - Value written in bits 29:24 of Work Register
		101 - Value written in bits 35:30 of Work Register
		110 - Value written in bits 41:36 of Work Register
		111 - Value written in bits 47:42 of Work Register
	ADPTV_OFFSET when SHFTVAL_SRC = 1 & FFT_RND = 0	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained
		000 - Value comes from bits 5:0 of WR
		001 - Value comes from bits 11:6 of WR
		010 - Value comes from bits 17:12 of WR
		011 - Value comes from bits 23:18 of WR
		100 - Value comes from bits 29:24 of WR
		101 - Value comes from bits 35:30 of WR
		110 - Value comes from bits 41:36 of WR
		111 - Value comes from bits 47:42 of WR
[24:16]	MCA_INC	Multiply Coefficient Address Increment, Address offset from coefficient to coefficient
[15:10]	ADPTV_WRNUM when SHFTVAL_SRC = 0 & FFT_RND > 0	Gives the Work Register index where the accumulated Block Shift value is written at the end of every fft round

Table continues on the next page...

Table 45-47. IRDX4 Instruction bit description (continued)

Bits	Bit Field	Description
	ADPTV_WRNUM when SHFTVAL_SRC = 1 & FFT_RND = 0	Gives the index of Work Register from where the pre-scaling left shift value is obtained
[9:8]	MULT_MOD	Mult Mode
		00 - Mode 0 - multiply with a single 32-bit complex immediate value
		01- Mode 1 - multiply with a single 32-bit complex constant from coefficient address (twiddle RAM address or work register index).
		10- Mode 2 - increment and modulus applied after every Radix4 step. coeff_addr = coeff_base_addr + ((step*MCA_INC) % MCA_MOD) * 8 + m where step denotes the step number. This step number = 0 in the first clock cycle, 1 in the second clock cycle, etc. MCA_INC = the address increment value MCA_MOD = modulo value m = 0 to 7, is the index of the address generated in a single step (out of the 8 addresses)
		11- Reserved
[7]	SHFTVAL_SRC	Gives the source from where the pre-scaling left shift value is obtained. In case WIN_TYPE > 0 this field should be SET only if MULT_MOD = 0, else there is an error indication.
		0 - Value comes from the command word
		1 - Value comes from Work Register
[6:4]	SHFT_VAL	Pre-scaling left shift value
		000 – no shift
		001 – shift left 1 bit
		010 - shift left 2 bits
		011 - shift left 3 bits
		100 - shift left 4 bits
		101 to 111 - shift left 8 bits
[2:0]	TW_OVS	Twiddle Oversampling - to use stored Wn with n > k*vector length
		0000 – no oversampling
		0001 – oversampling factor of 2
		0010 – oversampling factor of 4
		0011 – oversampling factor of 8
		0100 – oversampling factor of 16
		0101 - oversampling factor of 32
		0110 – oversampling factor of 64
		0111 – oversampling factor of 128
		1000 – oversampling factor of 256
		1001 to 1111 - oversampling factor of 512

Inverse Radix2 Instruction

The table below shows the instruction bit encoding used for Inverse Radix2(IRDX2) Command. The details of each bit field is defined in the sections below.

Table 45-48. IRDX2 Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(101100)						IN_DATTYP	Reserved		FFT_RND			Reserved	ADPTV	ADPTV_SHFT	QUAD_EXT
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
REPEAT		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ADPTV_OFFSET			Reserved								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADPTV_WRNUM						Reserved		SHFTVAL_SRC	SHFT_VAL			TW_OVS			

Table 45-49. IRDX2 Instruction bit description

Bits	Bit Field	Description
[121]	IN_DATTYP	Input data type
		0 - Real
		1 - Complex
[118:116]	FFT_RND	FFT Round - Last rounds only as mentioned
		000 - Reserved
		001 - Round 1 (for 8 point FFT)
		010- Round 2 (for 32 point FFT)
		011- Round 3 (for 128 point FFT)
		100- Round 4 (for 512 point FFT)
		101- Round 5 (for 2048 point FFT)
		110 and 111 - Reserved
[114]	ADPTV	0 - Adaptive scaling not enabled
		1 - Adaptive scaling enabled

Table continues on the next page...

Table 45-49. IRDX2 Instruction bit description (continued)

Bits	Bit Field	Description
[113]	ADPTV_SHFT	0 - MSB occupation counter counts unoccupied bits from bit 15 downwards
		1 - MSB occupation counter counts unoccupied bits from bit 23 downwards
[112]	QUAD_EXT	Quadrature Extension
		0 – No quadrature extension
		1 – Quadrature extension used
[111:110]	REPEAT	Decides whether combined fft operation is enabled or not
		00 - No combined fft operation
		01 - Combined FFT8, for vector length/8 x FFTs
		10 - Reserved
		11 - Combined FFT32, for vector length/32 x FFTs
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	MULTI_COEF_ADD	Twiddle or Multiplier Coefficient address
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[27:25]	ADPTV_OFFSET when SHFTVAL_SRC = 0 & FFT_RND > 0	This offset value gives the location in a Work Register where the accumulated Block Shift value is written at the end of every fft round
		000 - Value written in bits 5:0 of Work Register
		001 - Value written in bits 11:6 of Work Register
		010 - Value written in bits 17:12 of Work Register
		011 - Value written in bits 23:18 of Work Register
		100 - Value written in bits 29:24 of Work Register
		101 - Value written in bits 35:30 of Work Register
		110 - Value written in bits 41:36 of Work Register
		111 - Value written in bits 47:42 of Work Register
	ADPTV_WRNUM when SHFTVAL_SRC = 1 & FFT_RND = 0	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained
		000 - Value comes from bits 5:0 of WR
		001 - Value comes from bits 11:6 of WR
		010 - Value comes from bits 17:12 of WR
		011 - Value comes from bits 23:18 of WR
		100 - Value comes from bits 29:24 of WR
		101 - Value comes from bits 35:30 of WR

Table continues on the next page...

Table 45-49. IRDX2 Instruction bit description (continued)

Bits	Bit Field	Description
		110 - Value comes from bits 41:36 of WR
		111 - Value comes from bits 47:42 of WR
[15:10]	ADPTV_WRNUM when SHFTVAL_SRC = 0 & FFT_RND > 0	Gives the Work Register index where the accumulated Block Shift value is written at the end of every fft round
	ADPTV_WRNUM when SHFTVAL_SRC = 1 & FFT_RND = 0	Gives the index of Work Register from where the pre-scaling left shift value is obtained
[7]	SHFTVAL_SRC	Gives the source from where the pre-scaling left shift value is obtained
		0 - Value comes from the command word
		1 - Value comes from Work Register
[6:4]	SHFT_VAL	Pre-scaling left shift value
		000 – no shift
		001 – shift left 1 bit
		010 - shift left 2 bits
		011 - shift left 3 bits
		100 - shift left 4 bits
		101 to 111 - shift left 8 bits
[3:0]	TW_OVS	Twiddle Oversampling - to use stored Wn with n > k*vector length
		0000 – no oversampling
		0001 – oversampling factor of 2
		0010 – oversampling factor of 4
		0011 - oversampling factor of 8
		0100 - oversampling factor of 16
		0101 - oversampling factor of 32
		0110 - oversampling factor of 64
		0111 - oversampling factor of 128
		1000 - oversampling factor of 256
		1001 to 1111 - oversampling factor of 512

SCP Instruction

The table below shows the instruction bit encoding used for SCP Command. The details of each bit field is defined in the sections below.

Table 45-50. SCP Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112

Table continues on the next page...

Table 45-50. SCP Instruction (continued)

OPCODE(101010)						IN_DAT_TYP	RE_IM_COEFF		Reserved			NO_OF_TAPS			
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
Reserved		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
COEF_ADD															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				SHFTVAL_OFST			Reserved								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHFTVAL_WR_NO						Reserved		SHFTVAL_SRC	SHFT_VAL			Reserved			

Table 45-51. SCP Instruction bit description

Bits	Bit Field	Description
[121]	IN_DAT_TYP	Input data type
		0- Real data
		1 - Complex data
[120:119]	RE_IM_COEFF	Indicates whether coefficients are real, imaginary or complex
		00 and 01 - coefficients are complex
		10 - coefficients are imaginary OR coefficients are real but stored in the imaginary portion of the memory locations
		11 - coefficients are real and stored in the real portion of the memory locations
[115:112]	NO_OF_TAPS	Gives the number of coefficients in the Scalar product
		0000 to 0011 - 2 coefficients in scp
		0100 - 4 coefficients in scp
		0101 to 0111 - 2 coefficients in scp
		1000 - 8 coefficients in scp
		1001 to 1111 - 2 coefficients in scp
[109]	IMA	Indirect memory addressing

Table continues on the next page...

Table 45-51. SCP Instruction bit description (continued)

Bits	Bit Field	Description
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:98]	VEC_SZ	Vector size/length, should be a multiple of 8
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:48]	COEFF_ADD	Gives the twiddle RAM address where the coefficients for the scalar product are stored
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[27:25]	SHFTVAL_OFST	This field gives the offset in a Work Register from where shift value for pre-scaling is obtained
		000- Value comes from bits 5:0 of WR
		001- Value comes from bits 11:6 of WR
		010- Value comes from bits 17:12 of WR
		011- Value comes from bits 23:18 of WR
		100- Value comes from bits 29:24 of WR
		101- Value comes from bits 35:30 of WR
		110- Value comes from bits 41:36 of WR
		111- Value comes from bits 47:42 of WR
[15:10]	SHFTVAL_WR_NO	If SHFTVAL_SRC is set, then the this field indicates the index of Work Register that gives the shift value
[7]	SHFTVAL_SRC	Gives the source from which value of shift for pre-scaling is obtained
		1- Shift value comes from Work Register
		0- Shift value comes from command word
[6:4]	SHFT_VAL	Pre-scaling before multiplication
		000 - No shift
		001 - Left shift by 1 bit
		010 - Left shift by 2 bits
		011 - Left shift by 3 bits
		100 - Left shift by 4 bits
		101 to 111 - Left shift by 8 bits

45.9.8 COPY

45.9.8.1 Introduction

The copy operation primarily moves N values from one address location to another one. The operation works with 8 operands simultaneously. Many different flavours of the COPY operation are supported. These are described in the following sections.

45.9.8.2 Principle of Operation

The basic copy operation simply copies N vector elements from source address to the destination address. In case of packed operands with REAL or LOG2 data type the number of clock cycles taken by the copy operation is half as compared to a case of unpacked or COMPLEX operands. Many variations and options are available with the COPY hardware accelerator.

It is also possible to copy data from Operand RAM to Twiddle RAM and vice-versa. When copying data from Operand RAM to Twiddle RAM, 16 LSB bits of both real and imaginary operands get copied. Rest of the bits are ignored. This is shown in the figure below.

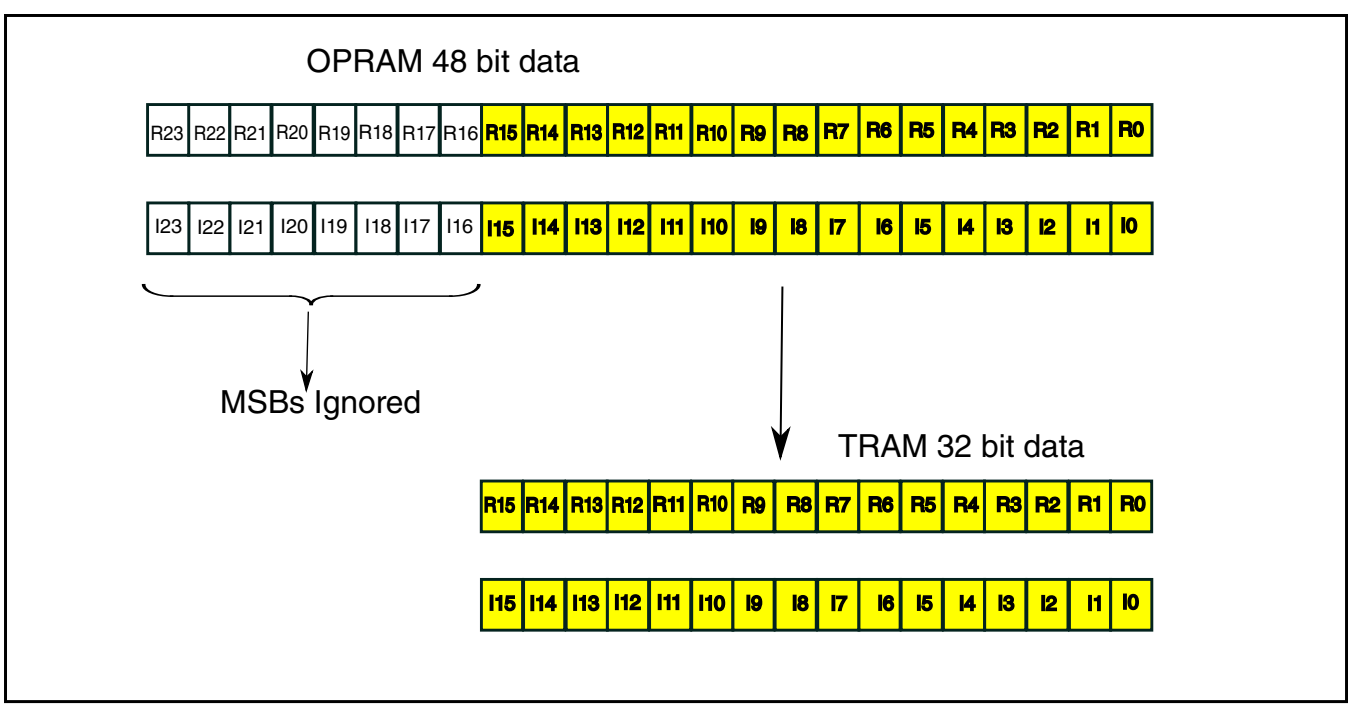


Figure 45-37. OPRAM to TRAM copy

When data is copied from Twiddle RAM to Operand RAM the MSB bit of both real and imaginary operands get extended to fill up the MSB locations. This is shown in the figure below.

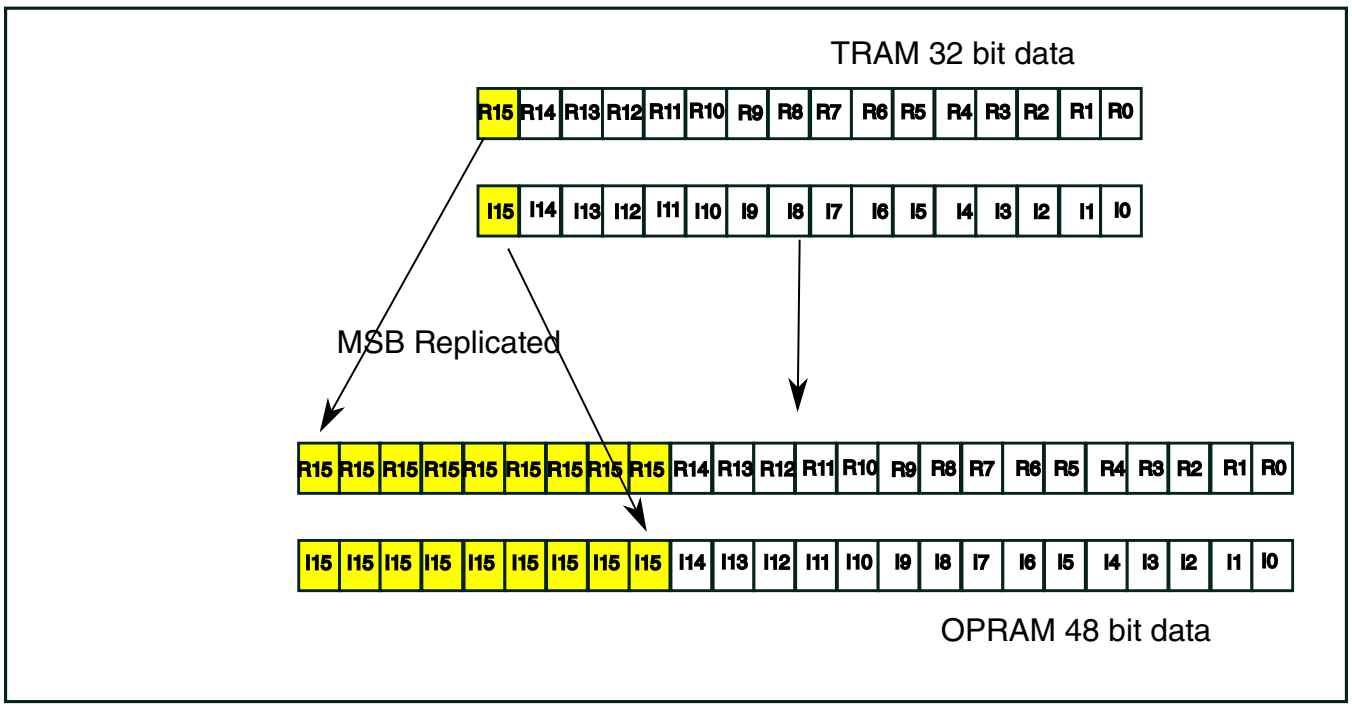


Figure 45-38. TRAM to OPRAM copy

Source and Destination addresses must be a multiple of 8 for all copy types other than Copy Shift operation. In Copy Shift operation the destination address cannot be a multiple of 8.

The different types of copy operations are described in the section below.

45.9.8.2.1 Simple Copy

Simple copy - A simple copy operation copies N elements, where N = VEC_SZ, from the address pointed to by the SRC_ADD to the address pointed to by the DEST_ADD. The source and destination addresses may be incremented for every next read as per the SRC_ADD_INC and DEST_ADD_INC values.

45.9.8.2.2 Threshold Copy

Greater than or equal to threshold copy - A greater than or equal to threshold copy operation copies elements from the SRC_ADD greater than the threshold value, to the DEST_ADD. All other elements are set to 0. The threshold value is stored in the address pointed to by the THLD_ADD. Only one threshold value is used for the entire vector length. Figure 45-39 shows an example of greater than threshold copy.

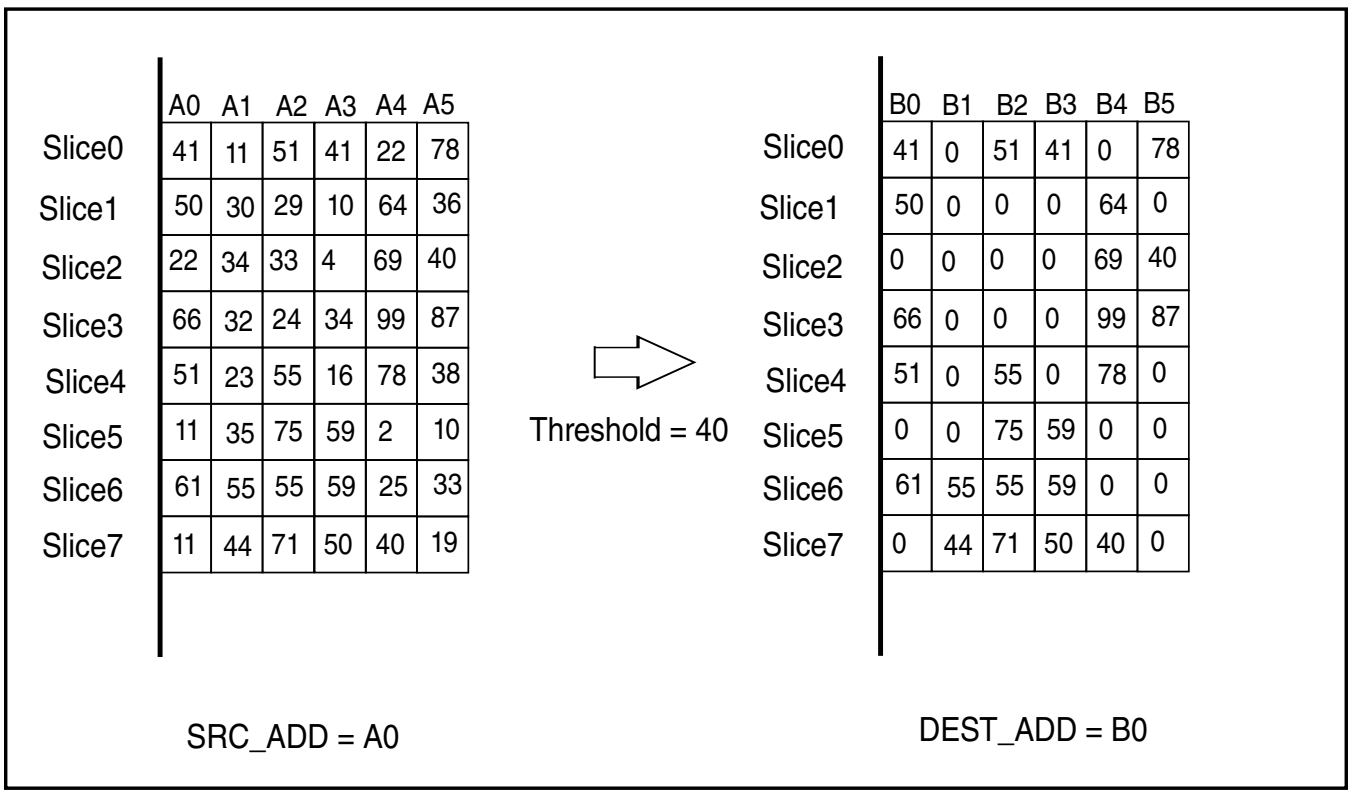


Figure 45-39. Greater than or equal to threshold copy

Less than threshold copy - A less than threshold copy operation copies elements from the SRC_ADD less than the threshold value, to the DEST_ADD. All other elements are set to 0. The threshold value is stored in the address pointed to by the THLD_ADD.

A count of all the elements reset to 0 is kept in WR_R0_RE[REAL_R0]. The RST_ACC bitfield of the instruction when set, resets the accumulated value in this register.

The threshold values are preprocessed values. No preprocessing is done on the threshold values. Only the real part of the THLD_ADD is used as the threshold value. It is advisable to keep the threshold values in one of the work registers. Keeping the threshold values in OPRAM or TRAM leads to lower performance.

45.9.8.2.3 Transpose Copy

Transpose copy - Transposition is a special copy operation that converts a column vector to a row vector and vice versa. Since in that case only 8 input operands are available concurrently, an intermediate buffer is required to perform this operation with sufficient performance. A transpose operation is only done on a matrix of 8x8. An option of block transpose is also provided in which multiple blocks of 8x8 are transposed. The transpose copy is only available with a vector of total length in multiples of 64. Number of 8x8 blocks to be transposed is given by VEC_SZ/64. Figure 45-40 shows as example of transpose copy operation.

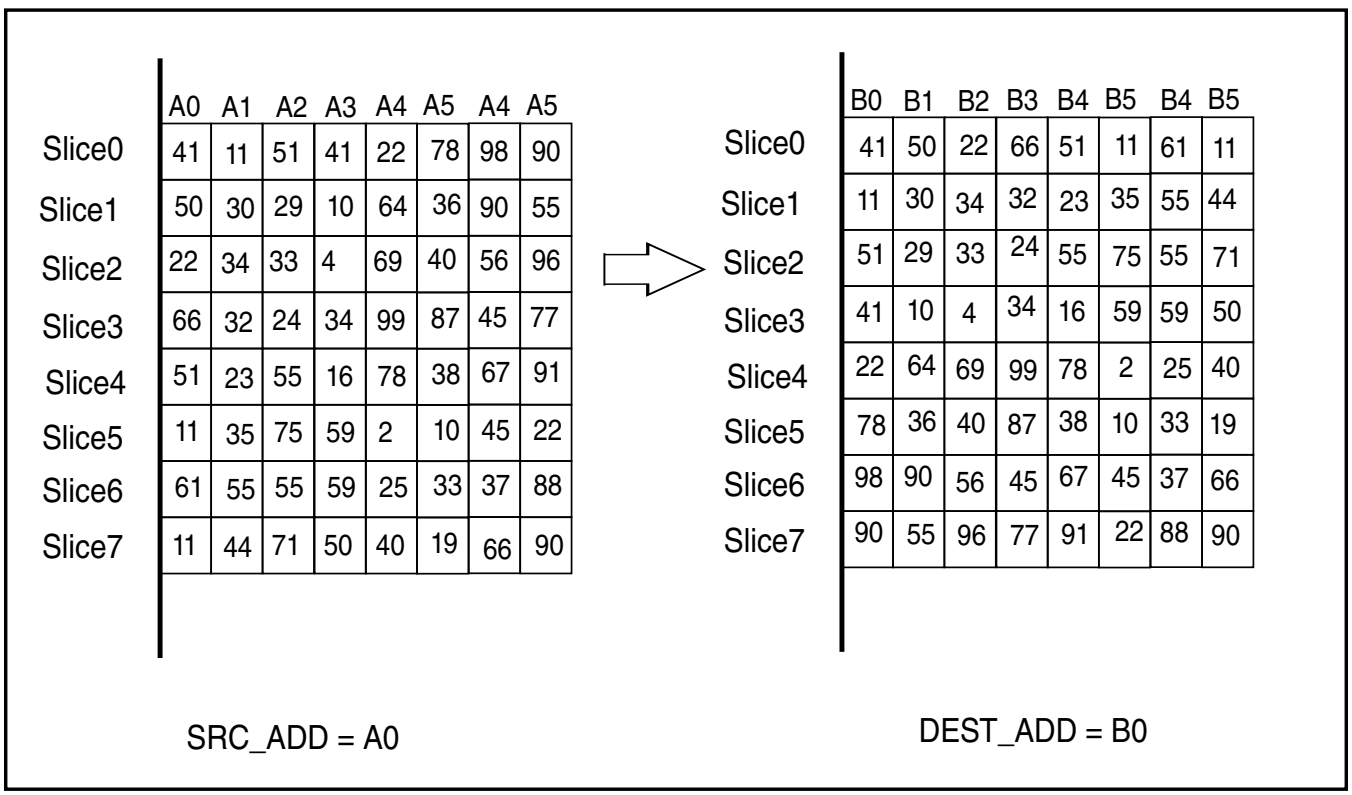


Figure 45-40. Transpose copy

45.9.8.2.4 Copy Real Pack

The copy real pack operation copies the real components from the input operands at the source address and packs them together at the destination address. The operands at the destination address after this operation are all real components in packed format. Figure 45-41 shows an example of copy real pack operation.

For copy real pack operation the VEC_SZ should be a multiple of 16.

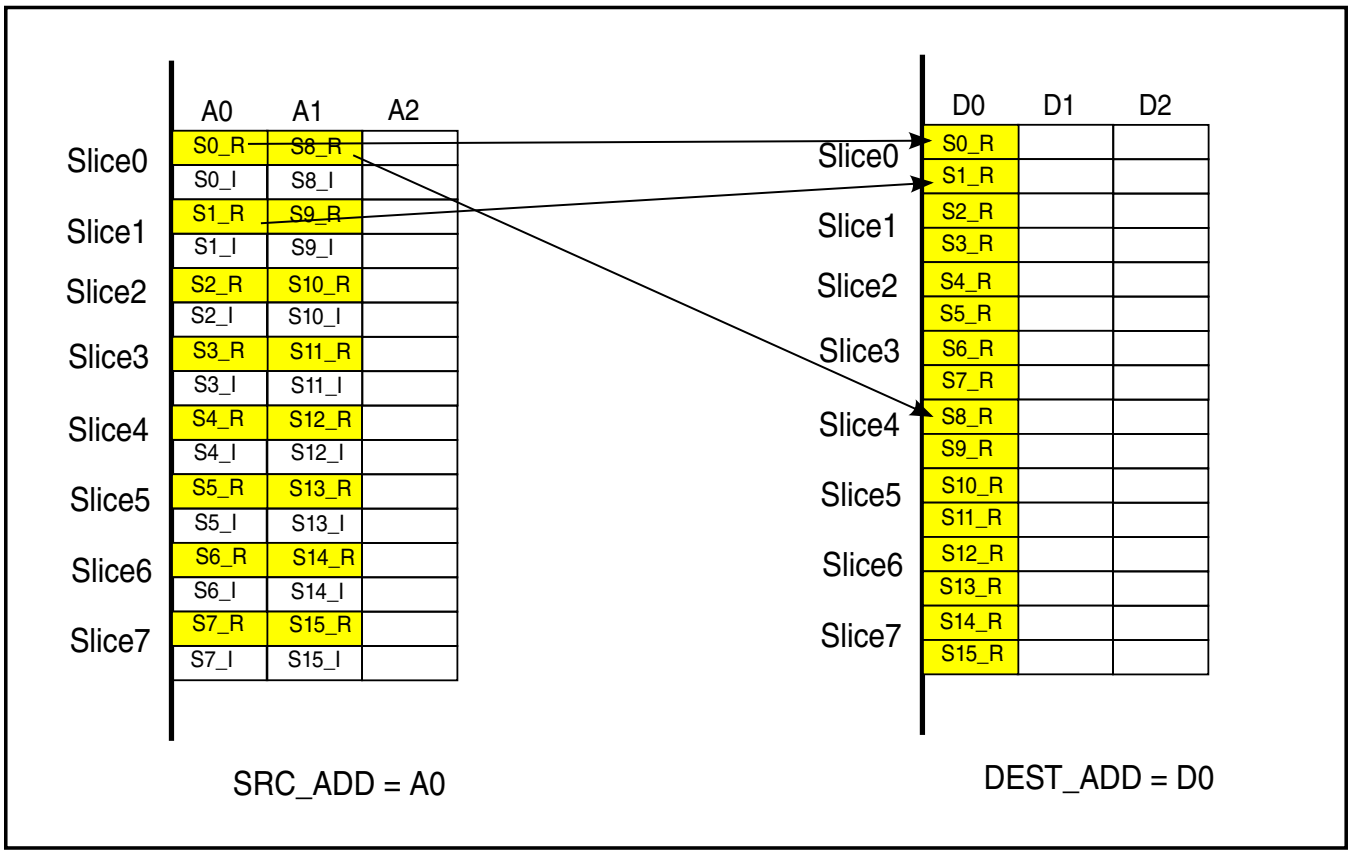


Figure 45-41. Copy real pack

45.9.8.2.5 Copy Imaginary Pack

The copy imaginary pack operation is similar to copy real pack operation except that instead of real components, imaginary components get copied to the destination address. The operands at the destination address after this operation are all imaginary components in packed format. Figure 45-42 shows an example of copy imaginary pack operation.

For copy imaginary pack, VEC_SZ should be a multiple of 16.

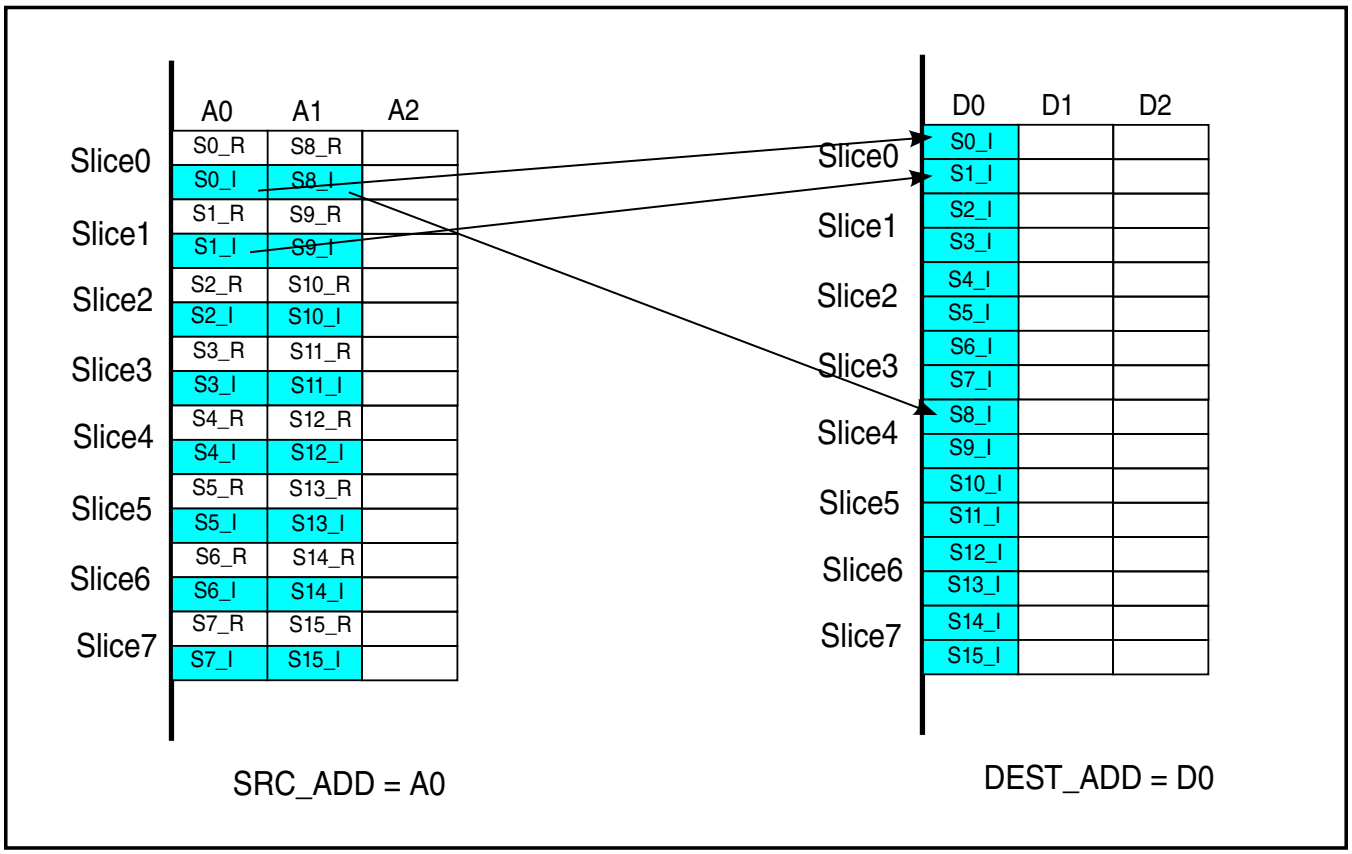


Figure 45-42. Copy imaginary pack

45.9.8.2.6 Copy Unpack

The copy unpack operation is an inverse of the copy pack operations. In the copy unpack operation the real and imaginary components at the source address are unpacked to the real components of the destination address. The imaginary components at the destination address may be kept as is or may be reset. This is decided by the RST_N_KEEP bit of the instruction set.

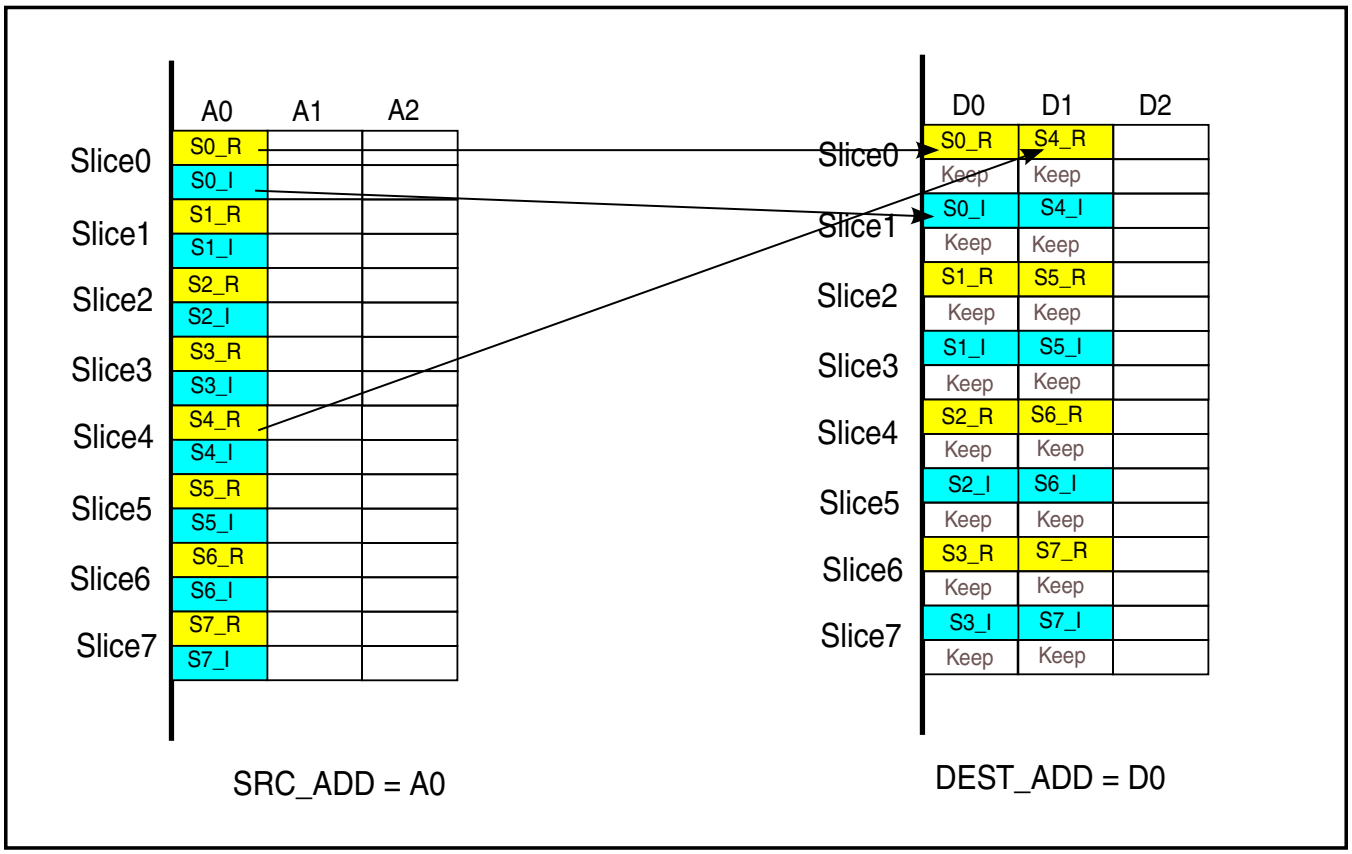


Figure 45-43. Copy unpack

45.9.8.2.7 Partial Copy Real

Partial copy real - The partial copy real operation copies only real operands from the source address to the destination address. The imaginary components at the destination may be kept as is or may be reset. This is decided by the RST_N_KEEP bit of the instruction set.

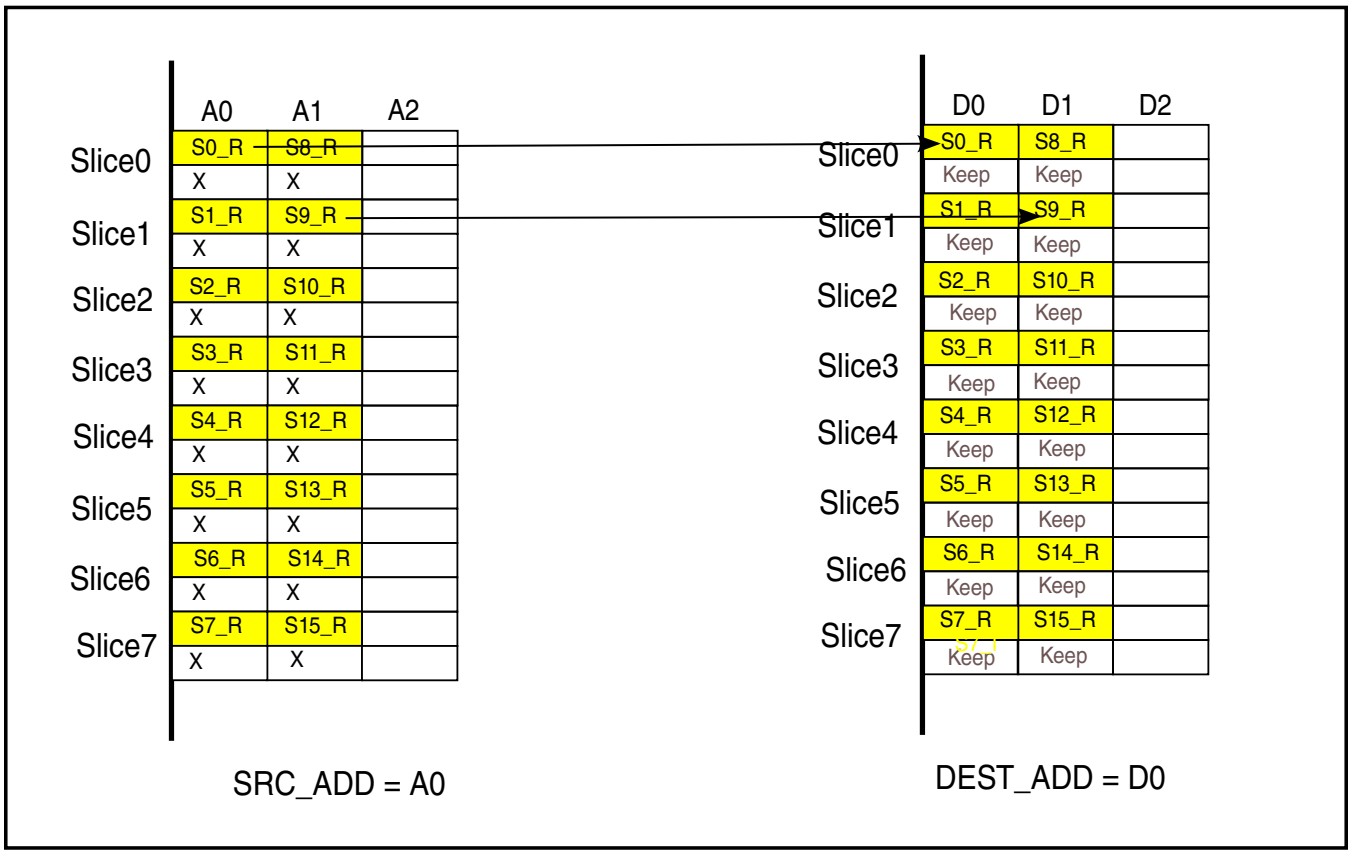


Figure 45-44. Partial copy real

45.9.8.2.8 Partial Copy Imaginary

The partial copy imaginary operation copies only imaginary operands from the source address to the destination address. The real components at the destination may be kept as is or may be reset. This is decided by the RST_N_KEEP bit of the instruction set.

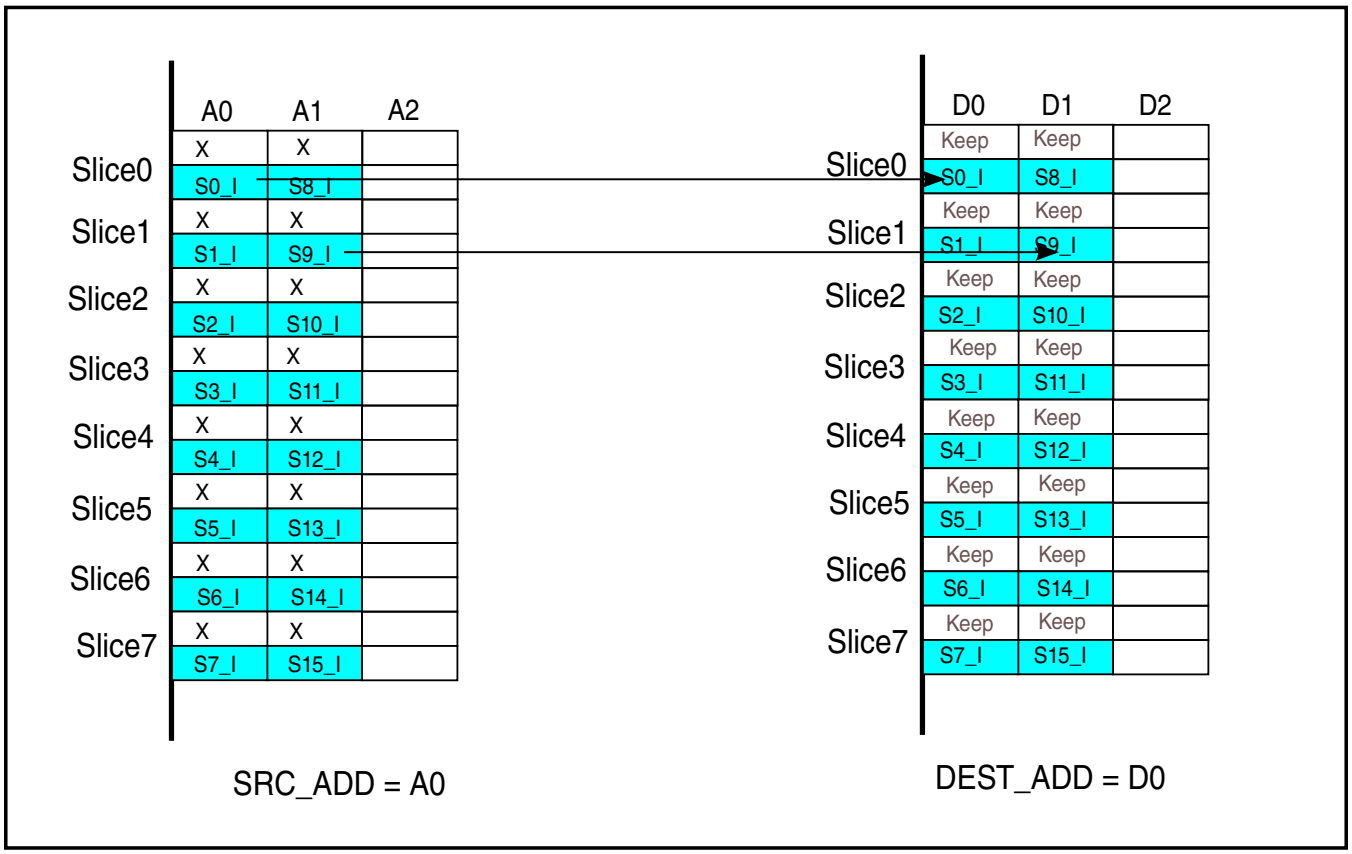


Figure 45-45. Partial copy imaginary

45.9.8.2.9 Partial Copy Real to Imaginary

The partial copy real to imaginary operation copies the real components at the source address to the imaginary components at the destination address. The real components at the destination may be kept as is or may be reset. This is decided by the RST_N_KEEP bit of the instruction set.

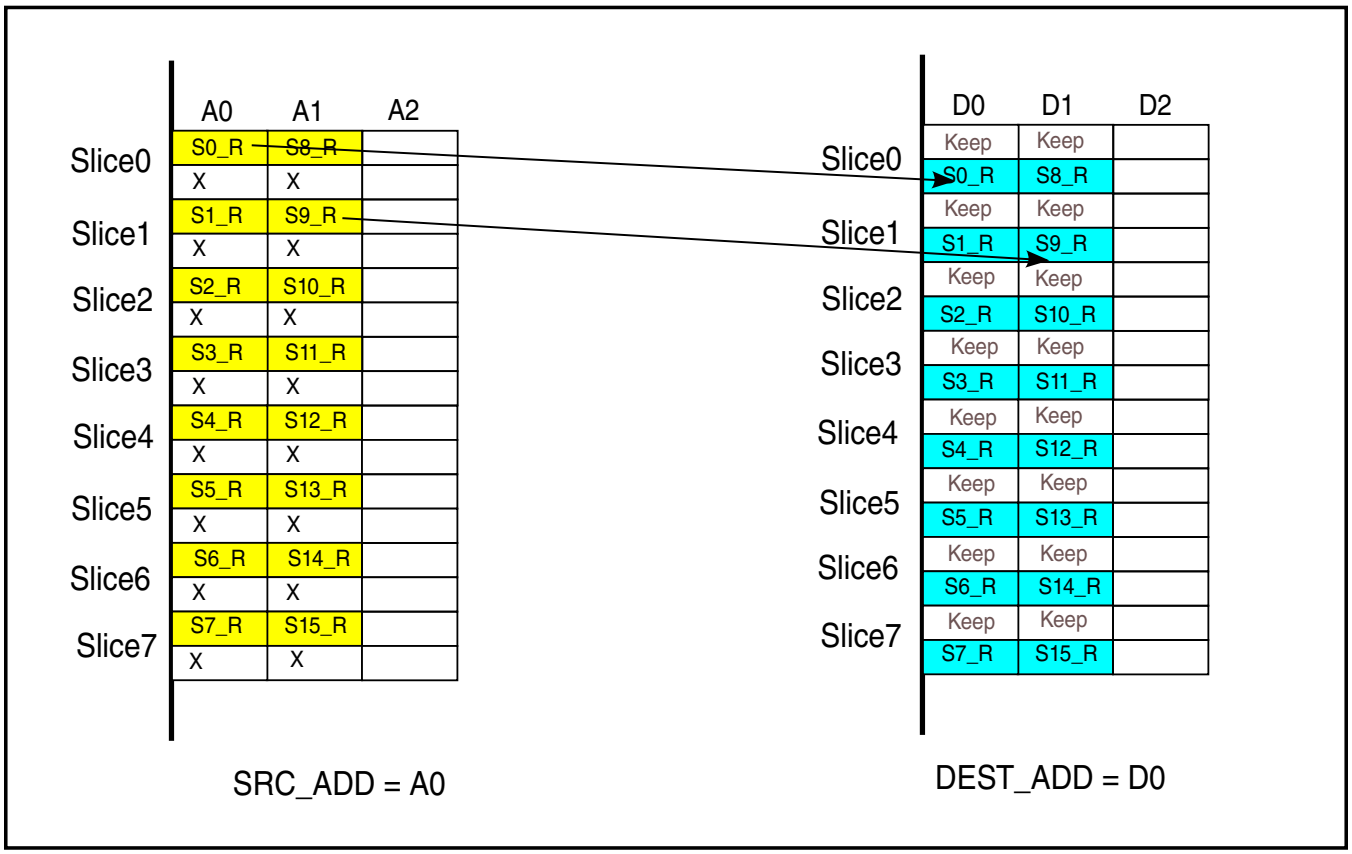


Figure 45-46. Partial copy real to imaginary

45.9.8.2.10 Partial Copy Imaginary to Real

The partial copy imaginary to real operation copies the imaginary components at the source address to the real components at the destination address. The imaginary components at the destination may be kept as is or may be reset. This is decided by the RST_N_KEEP bit of the instruction set.

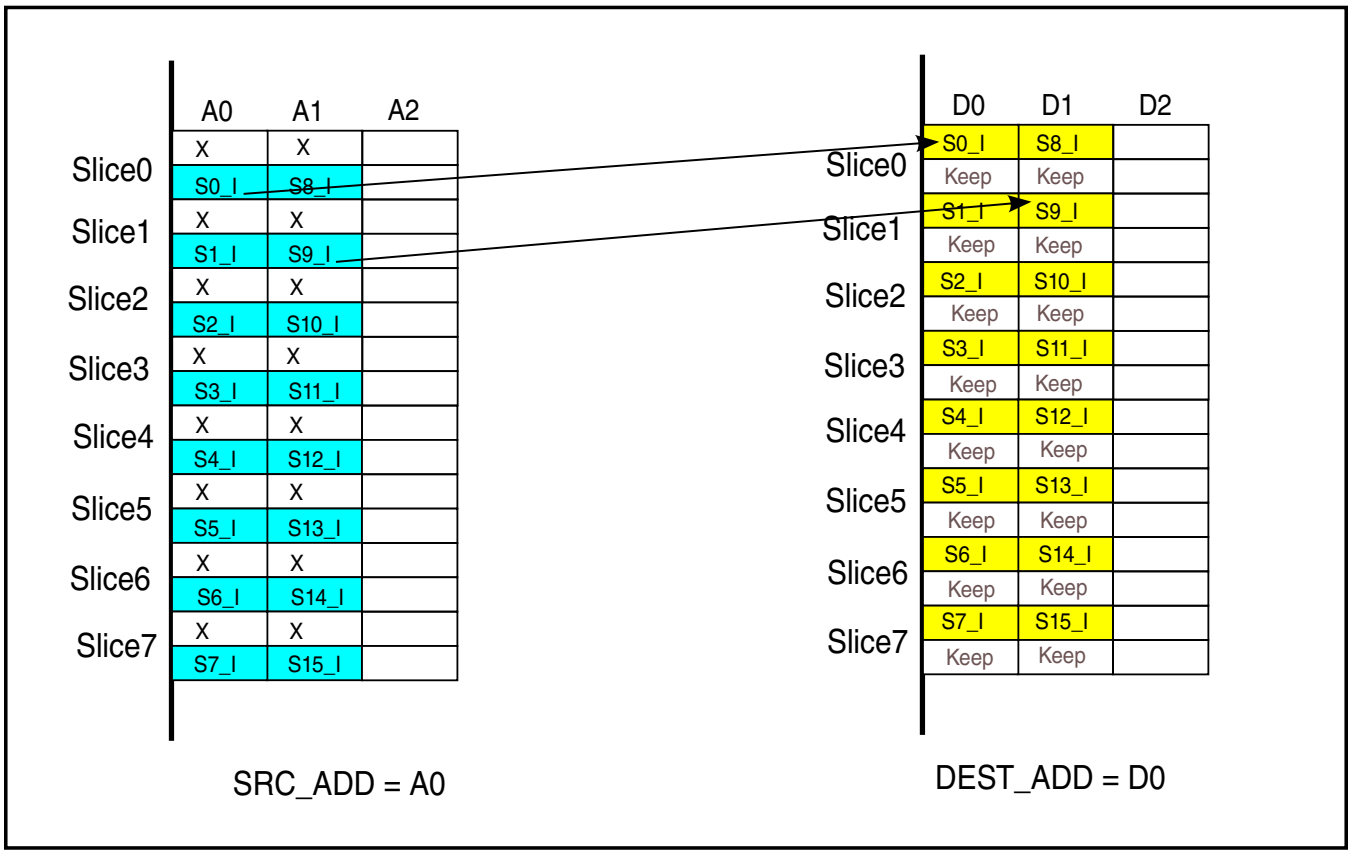


Figure 45-47. Partial copy imaginary to real

45.9.8.2.11 Clear

A clear operation simply overwrites certain elements at the DEST_ADD to 0. The SRC_ADD bitfield in this case is ignored by the controller. Which elements should be overwritten is determined by the MASK bitfield. The MASK bitfield is a 16 bit value where each bit stands for a real or imaginary component. A '1' in any location implies that the particular component is to be cleared. A '0' implies that the particular element is to be kept as is.

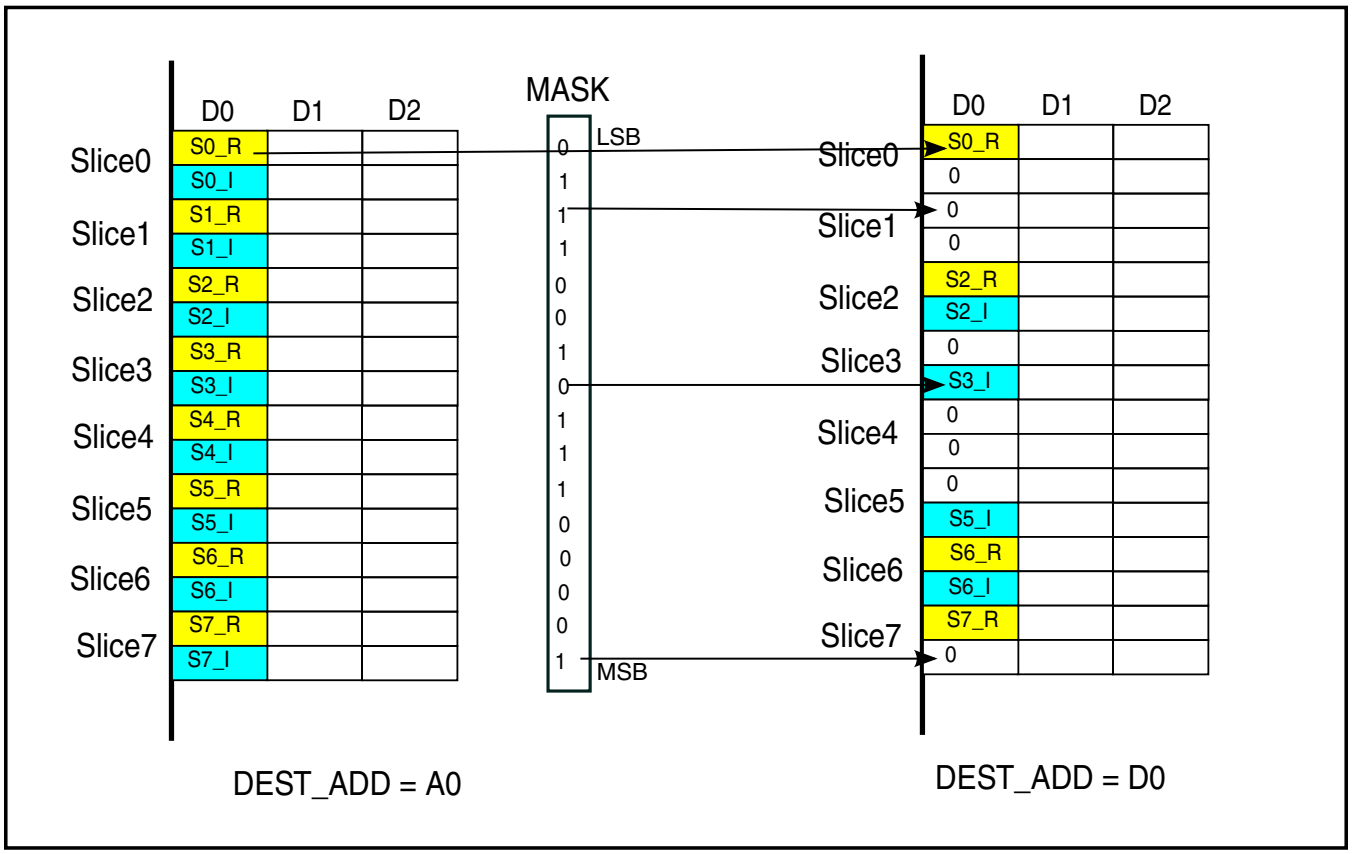


Figure 45-48. Copy clear with mask

45.9.8.2.12 Copy Shift

The copy shift operation copies the input operands at address SRC_ADD and copies them to destination address DEST_ADD just like simple copy operation. However, shift copy allows unaligned destination addresses. Therefore, the last three address bits of DEST_ADD defines the amount of shift. Figure 45-49 shows an example of copy shift operation. It is illegal for a copy shift operation to have an 8 aligned destination address (last three address bits 0). When such a configuration is selected, SPT_HW_ACC_ERR[COPY_IP_CMD_ERR] is set.

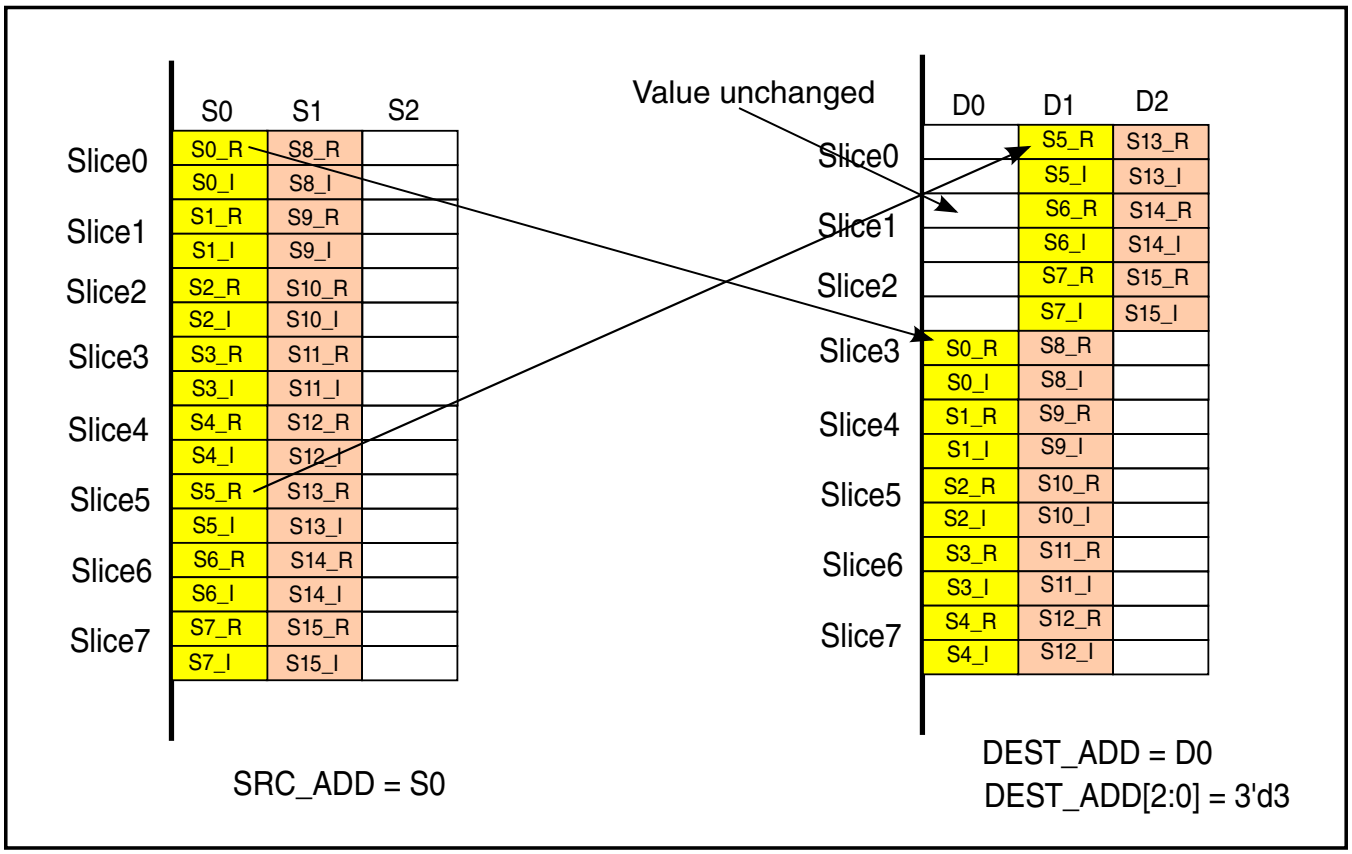


Figure 45-49. Copy Shift

45.9.8.2.13 8x4 Transpose Copy

8x4 Transpose copy - Transposition is a special copy operation that converts a column vector to a row vector and vice versa. Since in that case only 8 input operands are available concurrently, an intermediate buffer is required to perform this operation with sufficient performance. This unaligned transpose operation is done on a matrix of 8x4 (forward) or 4x8 (backward). For forward transpose, one column of 8 is converted into two rows of 4, and vice versa for backward transpose. An option of block transpose is also provided in which multiple blocks of 8x8 are transposed. The transpose copy is only available with a vector of total length in multiples of 64. Number of 8x8 blocks to be transposed is given by VEC_SZ/64. Figure 45-50 shows as example of both forward and backward transpose copy operation.

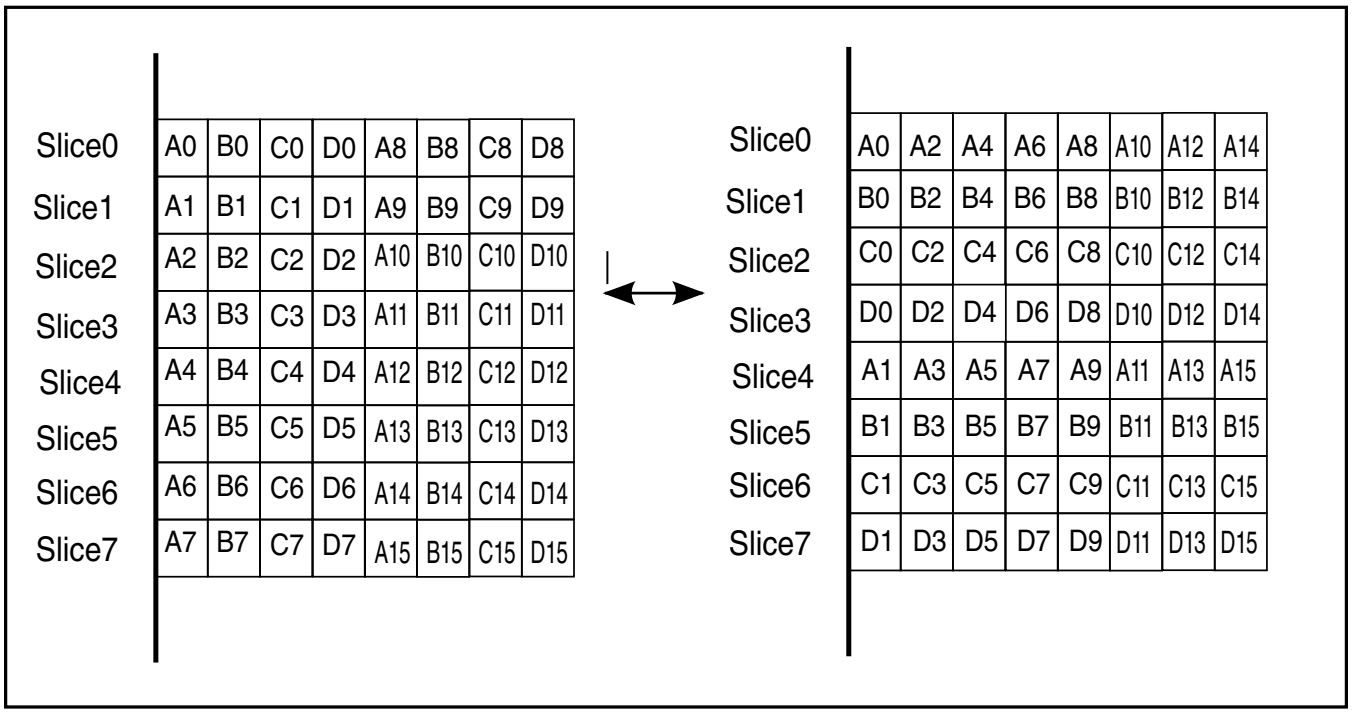


Figure 45-50. 8x4 Transpose copy

45.9.8.3 Copy Instruction Format

The table below shows the instruction bit encoding used for COPY. The details of each bit field and their operation is defined in the sections below.

Table 45-52. COPY Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100101)						IN_DATTYP		PREPROC		RST_ ACC	Reser ved	CP_TYPE			
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RST_ N_KE EP	Reser ved	IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
BLK_SRC_INC								BLK_DEST_INC							
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table 45-52. COPY Instruction (continued)

MASK															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THLD_ADD															
Reserved															

Table 45-53. COPY Instruction bit description

Bits	Bit Field	Description
[121:120]	IN_DATTYP	Input data type
		00 - Real
		01 - Complex
		10 - Log2
		11 - Reserved
[119:118]	PREPROC	Pre-processing (only valid for threshold copy)
		00 - No pre-processing (for log2 and others)
		01 - ABS(real) + ABS(im) (for real and complex numbers)
		10 - Magnitude for complex values (calculate half of approximate square root of real^2 + im^2 - only valid for complex numbers)
		11 - Reserved
[117]	RST_ACC	Reset accumulate (only valid for threshold copy)
		0 - No Reset operation
		1 - Resets real part of R0 Work Register : WR_R0_RE[REAL_R0]
[115:112]	CP_TYPE	Copy type
		0000 - Simple copy
		0001 - Threshold copy with >= option
		0010 - Threshold copy with < option
		0011 - Transpose copy
		0100 - Copy real pack
		0101 - Copy imaginary pack
		0110 - Copy unpack
		0111 - Partial copy real
		1000 - Partial copy imaginary
		1001 - Partial copy real to imaginary (R2I)
		1010 - Partial copy imaginary to real (I2R)
		1011 - Copy clear
		1100 - Copy shift
		1101 - Forward 8x4 transpose copy
		1110 - Backward 4x8 transpose copy
		1111 - Reserved
[111]	RST_N_KEEP	Reset values or keep original (valid for partial copies and copy unpack)

Table continues on the next page...

Table 45-53. COPY Instruction bit description (continued)

Bits	Bit Field	Description
		0 - Keep original values
		1 - Reset
[109]	IMA	Indirect memory addressing
		0 - Immediate memory addressing
		1 - Indirect memory addressing for source and destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source address
[79:64]	DEST_ADD	Destination address
[63:56]	BLK_SRC_INC	Block source address increment (valid for transpose copy operation with VEC_SZ > 64)
[54:48]	BLK_DEST_INC	Block destination address increment (valid for transpose copy operation with VEC_SZ > 64)
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[31:16]	MASK	Clear mask for clear operation
[15:0]	THLD_ADD	Threshold address

45.9.8.3.1 Block source and destination address increment

Section Transpose Copy describes the copy transpose operation. Copy transpose can work on multiple blocks of 8x8. An option is provided to increment the subsequent block addresses by the increment value. When the block source address increment value is set to a non-zero value, the address for the next 8x8 block fetch of operands is incremented by that value+1. Similarly, when the block destination address increment value is set to a non-zero value, the address of next block write to the memory is incremented by that value+1. By default when the block address increment value is set to 0, the address is incremented by 1 slice location. When the increment is a non-zero value, the address is incremented by (1+increment value) slice locations.

The block source and destination address increment fields are only valid when copy transpose operation is selected. For all other operations these fields are ignored.

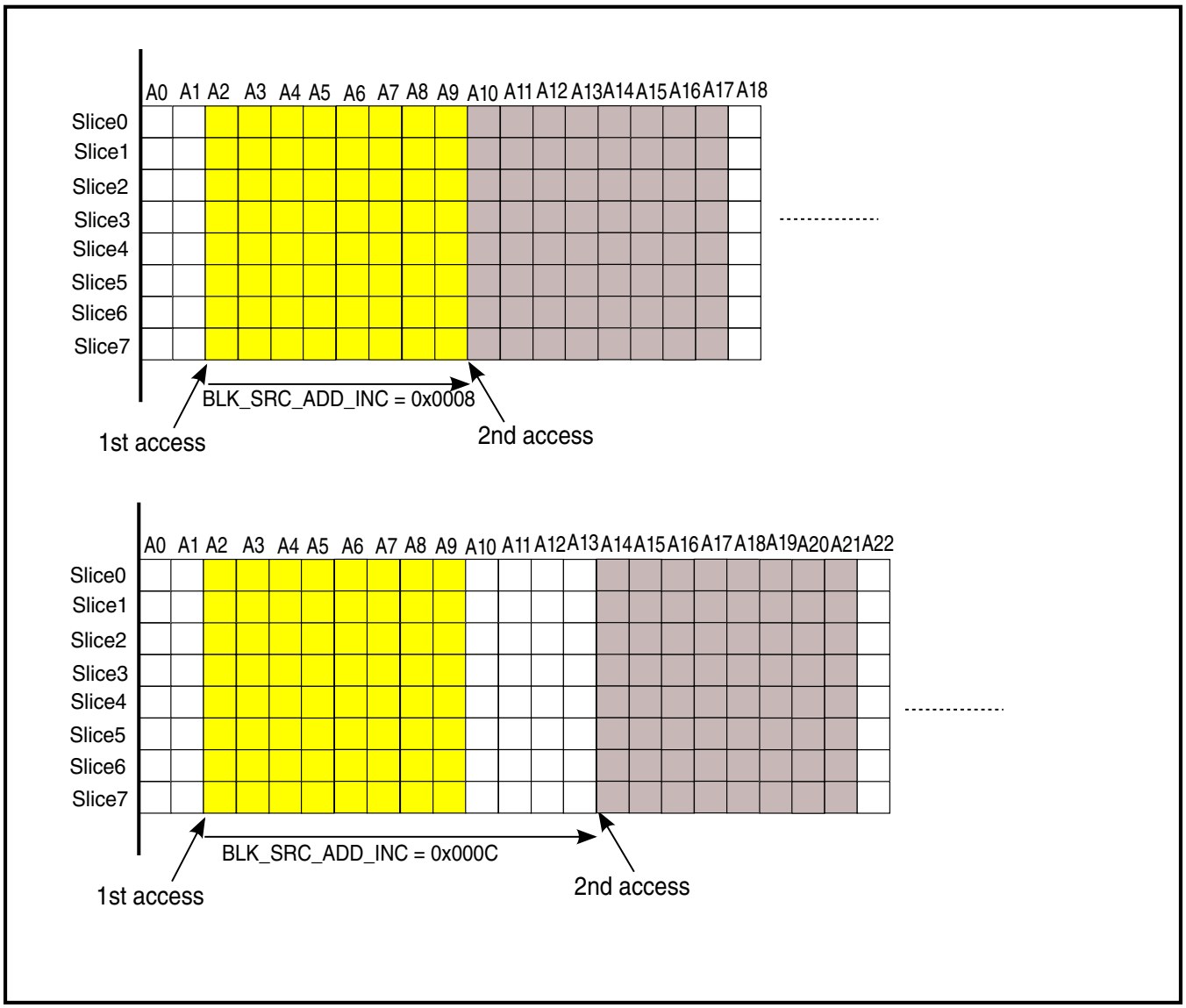


Figure 45-51. Block source address increment example

45.9.9 Programmable DMA

45.9.9.1 Introduction

The PDMA module is used to transfer the data from SPT internal memories (Twiddle RAM and Operand RAM) to system RAM and vice-versa. The PDMA transfer does not support data transfer to/from work register. The module is used to transfer the results of any mathematical operations on the data into the main system RAM via the System bus crossbar.

Transfer is initiated by the command from the sequencer. All the DMA control information comes from sequencer (i.e. input from command). The details of the command is provided in this chapter. The different fields in the command specify information such as the amount of data to be transferred, source and destination address of transfer.

PDMA module is capable of implementing various address schemes both for input and output. This gives software the flexibility to move data from source to destination seamlessly.

Apart from this, the module is capable of performing aggregation or compression/ decompression operation on-the-fly. This helps in reducing the data flow toward arbiter. The various compression schemes supported allow PDMA to compress the incoming data by various compression ratios thereby reducing the data traffic on the crossbar.

The PDMA is connected to the arbiter along with SDMA and command sequencer. The priority of access is highest for SDMA followed by PDMA and then the command sequencer.

All the register of the SPT blocks are implemented at common register interface. The status registers allow the PDMA to inform the software about any error condition encountered or the current number of words being transferred to the destination

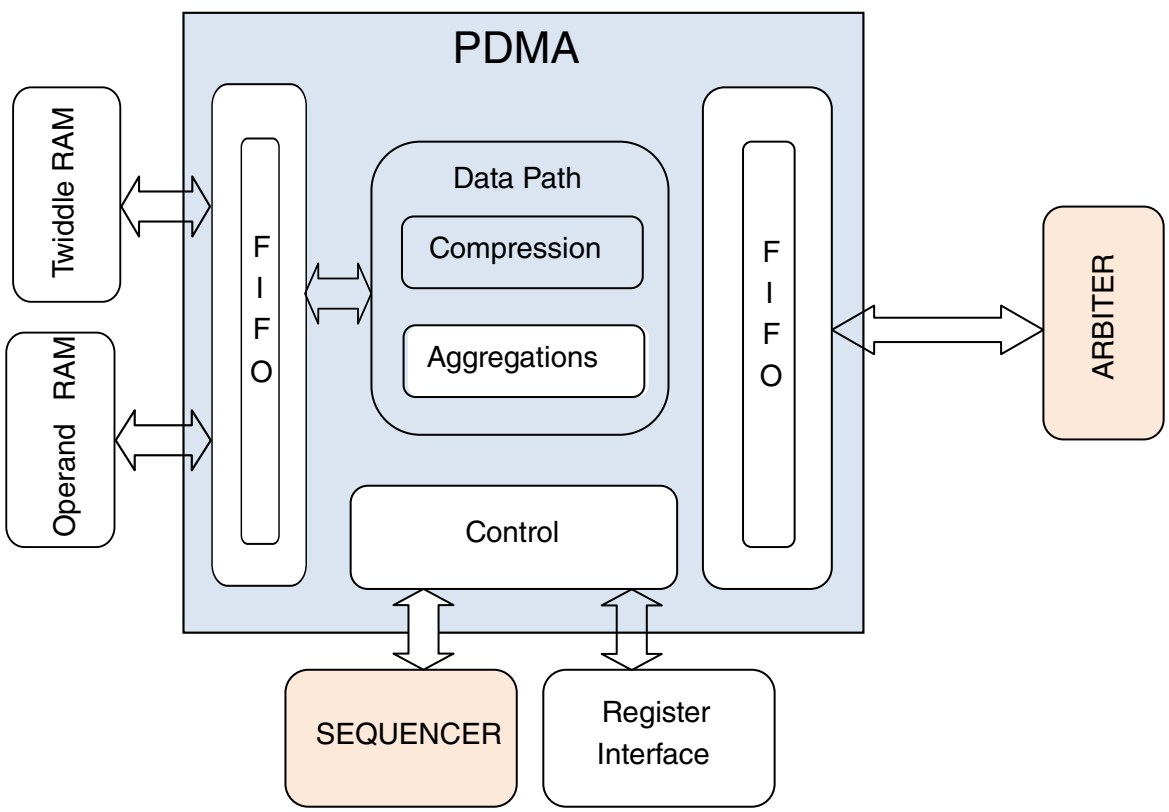


Figure 45-52. PDMA block diagram

45.9.9.2 Features

PDMA module has following features:

- PDMA data transfer is initiated by the sequencer and it provides the current status to the sequencer
  - One data transfer instruction is active at a time
- PDMA transfer is controllable from the sequencer input. It transfers the data between:
  - System RAM/Flash/TCM to Operand Ram or Twiddle Ram (SPT internal RAMs) and vice versa

  Note: PDMA cannot transfer the data between the twiddle ram and operand RAM
- Capable of handling 8 complex data in single cycle
- Data transfer size is controllable from the sequencer input
  - Maximum transfer Size from the twiddle ram is 4K
  - Maximum transfer size from the operand ram is 16K
  - Error if size exceeds the prescribed limit or size is 0
  - Transfer size is in terms of
    - Number of 64-bit words when transferring from System RAM to SPT/ Operand RAM's.
    - Number of 48-bit words when transferring from SPT/Operand RAM to System RAM.
    - Number of 32-bit words when transferring from Twiddle RAM to System RAM.
- Source and destination address is controllable from the sequencer input
- Addressing schemes
  - Skip ‘N’ address after ‘M’ address
  - M,N (for both source / destination) are input from the sequencer. These are in multiples of 8.

  Note: If the maximum address is exceeded, the address pointer rolls over.
- Separate control from the command for sign extension and 0 padding from the control
- Performs special packing and unpacking scheme on the fly.
  - 16-bit complex packing
  - 24-bit complex packing
  - 24-bit real packing
  - 16-bit real packing
  - 48-bit binary packing
  - 16-bit Mix Mode : Mode that combines two columns with 16-bit complex operands

    - CP6 compression
    - CP4 compression
    - CP8 compression
    - CP16 compression
    - CP4FMTA compression
    - CP4FMTB compression
    - CP8FMTB compression
    - CP16FMTB compression
- Separate control for sign extension or zero padding for all the packing/unpacking schemes
- Rounding of mantissa (during compression) is done by adding a random number (implemented using LFSR) or fixed value 'h80
- Compression algorithm
    - LFSR used for rounding off in format conversion
    - CP4, CP6, CP8, CP16, CP4FMTA, CP4FMTB, CP8FMTB, CP16FMTB compression and de-compression (except for CP16)
- Supports aggregation
    - Absolute
    - Index Value
    - Index cross sum

  Note: If aggregation is enabled then Packing/ Unpacking scheme doesn't have any effect on the data
- Keeps count of number of bytes transferred
    - Resets on every new transfer initiated
    - Updates signal provided at the end of transfer for register to store
- Generates errors
    - DMA command error : These are generated if the following conditions are encountered in the PDMA instruction
        - System and SPT/Operand memory start address is not in multiple of 8
        - Vector length is not a multiple of 8 for any SPTRAM-SRAM transfer
        - Vector length is not a multiple of 16 for 16 bit Mix Mode SPTRAM-SRAM transfer
        - Vector length is not a multiple of 4 for 16 bit complex pack SRAM-SPTRAM transfer
        - Vector length is not a multiple of 4 for 16 bit complex swap pack SRAM-SPTRAM transfer
        - Vector length is not a multiple of 8 for 24 bit complex pack SRAM-SPTRAM transfer
        - Vector length is not a multiple of 4 for 24 bit Real pack SRAM-SPTRAM transfer

- Vector length is not a multiple of 2 for 16 bit Real pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 8 for 16 bit Mix Mode pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 8 for 48 bit binary pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 2 for CP4D pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 16 for CP16 pack SPTRAM-SRAM transfer
- Any memory continuous address is set as zero
- OPRAM Continuous Address should be in multiple of 2 for 16 bit Mix Mode
- vector length is set as zero
- Transfer initiated to or from work registers
- Aggregation Mode does not support decompression
- Incorrect data packing mode specified
- Higher vector length programmed than permissible.
- Vector length is not a multiple of 2 for CP4DFMTB pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 2 for CP8FMTB pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 4 for CP16FMTB pack SRAM-SPTRAM transfer
- Vector length is not a multiple of 16 for CP16FMTB pack SPTRAM-SRAM transfer
- SRAM continuous address is not power of 2 in case of CP4FMTB, CP4DFMTB, CP8FMTB, CP16FMTB

- iIlegal command error : It is generated if proprietary algorithms are used by the non-authorized users.

45.9.9.3 Data Packing/Unpacking

The PDMA supports various methods to pack/unpack data being transfered between SPT RAM/Twiddle RAM and System RAM. While transferring the data between memories the data can be adjusted to 64 bits controlled by Sign Extension bit in the PDMA instruction (SE).

SE means sign extension of the value to the right. Sign extension appends '0' bits to the left if the right value MSB is '0' and '1' if the right value MSB is '1'.The data that needs to be transfered can be in any of the formats. The various methods supported are as follows:

- 16-bit packing: This packs four 16-bit complex data in SPT RAM into one 64-bit data in System RAM
- 24-bit complex packing: This packs 24-bit real and 24-bit imaginary part stored in the SPT RAM is stored in 64-bit data in System RAM. Each data is sign extended with MSB bit replicated for the remaining 8 bits
- 24-bit real packing: This packs two 24-bit data into one 64-bit data in System RAM. The bits are sign extended to make these 32-bit each
- 16-bit real packing: This packs four 16-bit data into one 64-bit data
- 48-bit binary packing: Each 48-bit data in SPT RAM is converted to 64-bit in System RAM by appending zero's in the MSB.
- 16-bit Mix Mode: This mode combines two columns with 16 bit complex operands to 8x 64 bits and vice versa
- Compression Modes (CP4/CP6/CP8/CP16/CP4FMTA/CP4FMTB/CP8FMTB/ CP16FMTB) are special types of data packing schemes which reduce the amount of data that is written in System RAM.
- CP6 compression packing: This takes in 6 complex data (each having 24-bit real and 24-bit imaginary) into 64-bit output at the System RAM side.
- CP4 Compression packing: This takes in 4 complex data (each having 24-bit real and 24-bit imaginary) into 64-bit output at System RAM side.
- CP8 Compression packing: This takes in 8 complex data (each having 24-bit real and 24-bit imaginary) into 64-bit output at System RAM side.
- CP16 compression packing: This takes in 16 complex data (each having 24-bit real and 24-bit imaginary) into 4 x 64 bit output at System RAM side.
- CP4FMTA Compression packing: This takes in 4 complex data (each having 24-bit real and 24-bit imaginary), compresses it into 32-bit value. Two 32-bit values are packed and stored as 64 bit output at System Ram side.
- CP4FMTB Compression packing: This takes in 4 complex data (each having 24-bit real and 24-bit imaginary), compresses it into 64-bit mantissa and 4 bit exponent and stores mantissa and exponent at System Ram side seperately starting from different offset.
- CP8FMTB Compression packing: This takes in 8 complex data (each having 24-bit real and 24-bit imaginary), compresses it into two 64-bit mantissas and 4 bit exponent and stores mantissa and exponent at System Ram side seperately starting from different offset.
- CP16FMTB Compression packing: This takes in 16 complex data (each having 24-bit real and 24-bit imaginary), compresses it into four 64-bit mantissas and 4 bit exponent and stores mantissa and exponent at System Ram side seperately starting from different offset.

NOTE

Assumption is that input is 24-bit real at LSB and 24-bit imaginary component at MSB. Compression would be done while transferring the data to System RAM and de-compression would be done while transferring the data to SPT/Twiddle RAM.

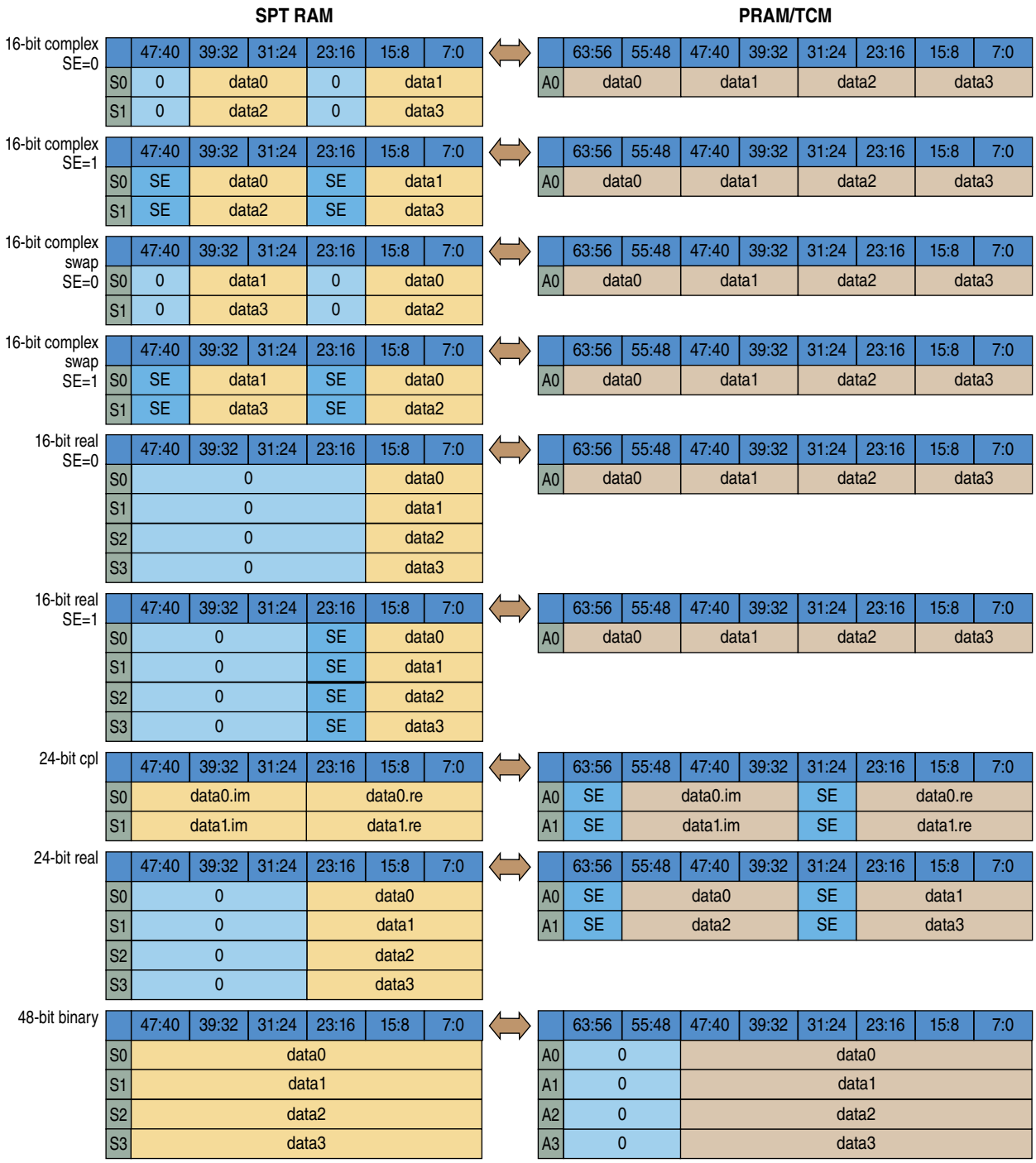


Figure 45-53. Data Format 1

	CA0						CA1					
	47:40	39:32	31:24	23:16	15:8	7:0	47:40	39:32	31:24	23:16	15:8	7:0
S0		A0.im			A0.re			B0.im			B0.re	
S1		A1.im			A1.re			B1.im			B1.re	
S2		A2.im			A2.re			B2.im			B2.re	
S3		A3.im			A3.re			B3.im			B3.re	
S4		A4.im			A4.re			B4.im			B4.re	
S5		A5.im			A5.re			B5.im			B5.re	
S6		A6.im			A6.re			B6.im			B6.re	
S7		A7.im			A7.re			B7.im			B7.re	

	63:48	47:32	31:16	15:0
A0	A0.im	A0.re	B0.im	B0.re
A1	A1.im	A1.re	B1.im	B1.re
A2	A2.im	A2.re	B2.im	B2.re
A3	A3.im	A3.re	B3.im	B3.re
A4	A4.im	A4.re	B4.im	B4.re
A5	A5.im	A5.re	B5.im	B5.re
A6	A6.im	A6.re	B6.im	B6.re
A7	A7.im	A7.re	B7.im	B7.re

Figure 45-54. Data Format 2

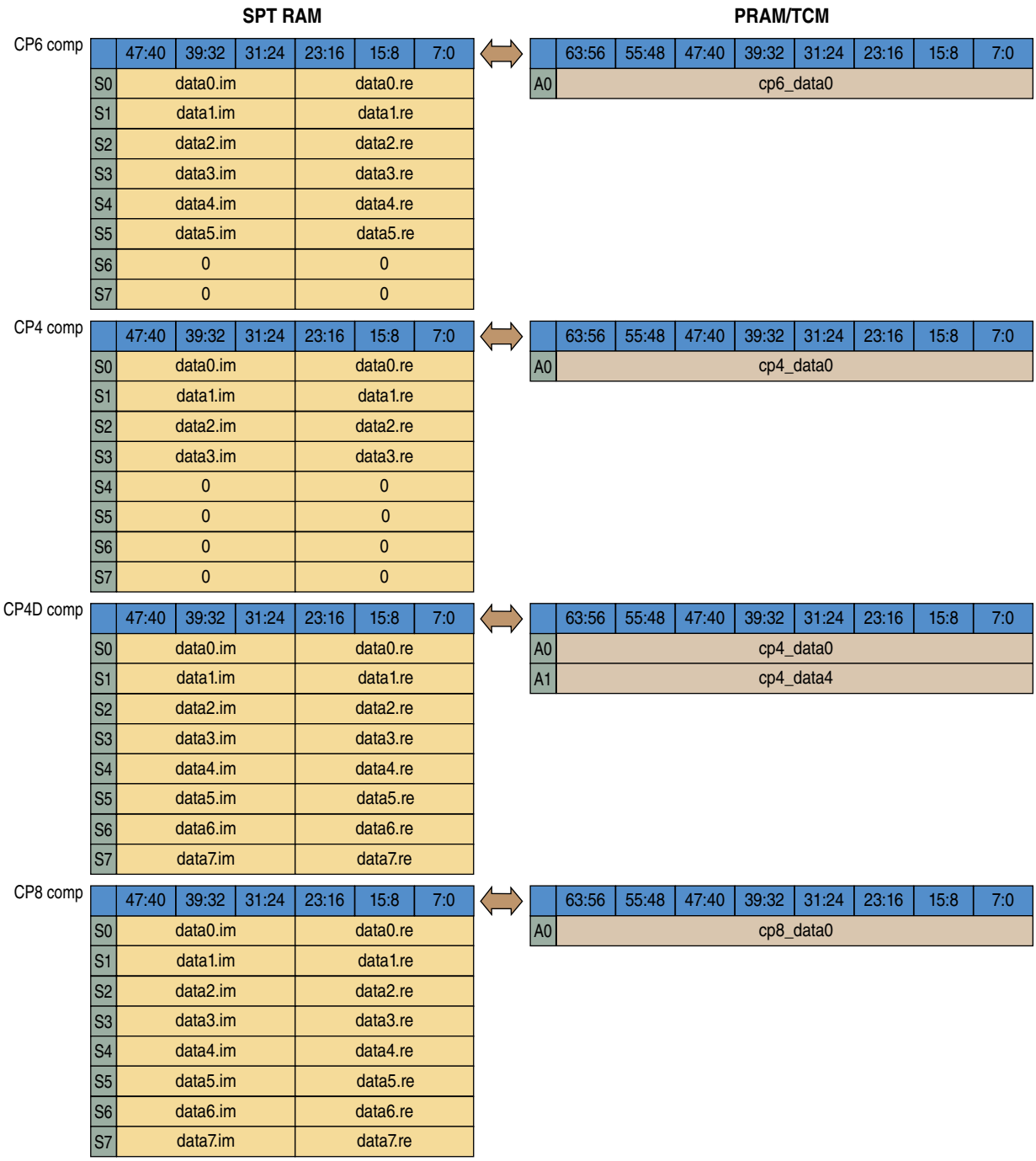


Figure 45-55. Data Format 3

CP4FMTA comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	0			0		
S5	0			0		
S6	0			0		
S7	0			0		
S0	data4.im			data4.re		
S1	data5.im			data5.re		
S2	data6.im			data6.re		
S3	data7.im			data7.re		
S4	0			0		
S5	0			0		
S6	0			0		
S7	0			0		

	63:60	59:56	55:28	27:0
A0	cp4_exp0	cp4_exp4	cp4_data0	cp4_data4

*For TRANS_TYPE 1 (compression): In case VECTOR_LEN is non multiple of 16, cp4_data4 is written as 'h0 and cp4_exp4 is written as 'hF for last data write.
For TRANS_TYPE 0 (decompression): Always complete 64 bit data is decompressed.

CP4DFMTA comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	data4.im			data4.re		
S5	data5.im			data5.re		
S6	data6.im			data6.re		
S7	data7.im			data7.re		

	63:60	59:56	55:28	27:0
A0	cp4_exp0	cp4_exp4	cp4_data0	cp4_data4

CP4FMTB comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	0			0		
S5	0			0		
S6	0			0		
S7	0			0		

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A0	d0_re	d0_im	d1_re	d1_im	d2_re	d2_im	d3_re	d3_im

	63:60	59:0
B0	exp0	

d*.im represents mantissa of data*.im
d*.re represents mantissa of data*.re
exp0 represents exponent for 4 complex data

CP4DFMTB comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	data4.im			data4.re		
S5	data5.im			data5.re		
S6	data6.im			data6.re		
S7	data7.im			data7.re		

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A0	d0_re	d0_im	d1_re	d1_im	d2_re	d2_im	d3_re	d3_im
A1	d4_re	d4_im	d5_re	d5_im	d6_re	d6_im	d7_re	d7_im

	63:60	59:56	55:0
B0	exp0	exp4	

d*.im represents mantissa of data*.im
d*.re represents mantissa of data*.re
exp0 represents exponent for first 4 complex data
exp4 represents exponent for next 4 complex data

Figure 45-56. Data Format 3

CP8FMTB comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	data4.im			data4.re		
S5	data5.im			data5.re		
S6	data6.im			data6.re		
S7	data7.im			data7.re		

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A0	d0_re	d0_im	d1_re	d1_im	d2_re	d2_im	d3_re	d3_im
A1	d4_re	d4_im	d5_re	d5_im	d6_re	d6_im	d7_re	d7_im

	63:60	59:0
B0	exp0	

d*.im represents mantissa of data*.im
d*.re represents mantissa of data*.re
exp0 represents exponent for 8 complex data

CP16FMTB comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	data4.im			data4.re		
S5	data5.im			data5.re		
S6	data6.im			data6.re		
S7	data7.im			data7.re		
S8	data8.im			data8.re		
S9	data9.im			data9.re		
S10	data10.im			data10.re		
S11	data11.im			data11.re		
S12	data12.im			data12.re		
S13	data13.im			data13.re		
S14	data14.im			data14.re		
S15	data15.im			data15.re		

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A0	d0_re	d0_im	d1_re	d1_im	d2_re	d2_im	d3_re	d3_im
A1	d4_re	d4_im	d5_re	d5_im	d6_re	d6_im	d7_re	d7_im
A2	d8_re	d8_im	d9_re	d9_im	d10_re	d10_im	d11_re	d11_im
A3	d12_re	d12_im	d13_re	d13_im	d14_re	d14_im	d15_re	d15_im

	63:60	59:0
B0	exp0	

d*.im represents mantissa of data*.im
d*.re represents mantissa of data*.re
exp0 represents exponent for 16 complex data

Figure 45-57. Data Format 3

SPT RAM **PRAM/TCM**

CP16 comp

	47:40	39:32	31:24	23:16	15:8	7:0
S0	data0.im			data0.re		
S1	data1.im			data1.re		
S2	data2.im			data2.re		
S3	data3.im			data3.re		
S4	data4.im			data4.re		
S5	data5.im			data5.re		
S6	data6.im			data6.re		
S7	data7.im			data7.re		
S0	data8.im			data8.re		
S1	data9.im			data9.re		
S2	data10.im			data10.re		
S3	data11.im			data11.re		
S4	data12.im			data12.re		
S5	data13.im			data13.re		
S6	data14.im			data14.re		
S7	data15.im			data15.re		

	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
A0	d0_re	d0_im	d1_re	d1_im	d2_re	d2_im	d3_re	d3_im
A1	d4_re	d4_im	d5_re	d5_im	d6_re	d6_im	d7_re	d7_im
A2	d8_re	d8_im	d9_re	d9_im	d10_re	d10_im	d11_re	d11_im
A3	d12_re	d12_im	d13_re	d13_im	d14_re	d14_im	d15_re	d15_im

d*.im represents mantissa of data*.im considering common exponent of all 16 operands.
d*.re represents mantissa of data*.re considering common exponent of all 16 operands.
CP16 does not have decompression.

Figure 45-58. Data Format 4

45.9.9.4 Linear Feedback Shift Register

Linear Feedback is a shift register whose input bit is a linear XOR function of its previous state. In our case the LFSR polynomial is:

$x^{64}+x^{61}+x^{34}+x^{9}+1$

LFSR is used only during the compression modes by the PDMA.

LFSR is controlled by the registers interface. The initial value to be loaded in LFSR can be set by software. This value gets loaded to the LFSR when the control bit (PDMA_LFSR_LOAD_EN) is set from the software. This bit automatically gets cleared by the design. PDMA should be in inactive state while programming this bit.

45.9.9.5 Compression

The PDMA includes a function for on-the fly data decompression and compression. It is a special case for packing/unpacking. The compression function is lossy and can only be applied to complex data. Depending on the compression mode, the module takes up to 4/6/8 complex operands and packs them into a single bit word thereby resulting in data compression.

Table 45-54. Compression type comparison

Compression Type	Input Data(48-bit complex)	Output Data(64-bit)	Compression Ratio
CP4	4	1	3
CP6	6	1	4.5
CP8	8	1	6
CP16	16	4	3
CP4FMTA	4	0.5	6
CP4FMTB	4x16	16+1	2.82
CP8FMTB	8x16	2x16+1	2.91
CP16FMTB	16x16	4x16+1	2.95

PDMA module supports the following compression modes:

1. **CP6 compression**

   The CP6 compression involves compressing 6 complex numbers(A-F) and storing them as a single 64-bit number. The compression assumes the 7th and 8th complex number is zero and ignores this for compression procedure as mentioned below.

   A0 B0 C0 D0 E0 F0 0 0

A1 B1 C1 D1 E1 F1 0 0

.....

.....

A7 B7 C7 D7 E7 F7 0 0 …

2. **CP4 compression**

   The CP4 compression involves compressing 4 complex numbers and storing them as a single 64-bit number.

   There are two different flavors for CP4. In this case the software shall store the numbers as follows

   A0 B0 C0 D0 0 0 0 0

   A1 B1 C1 D1 0 0 0 0

   .....

   .....

   A7 B7 C7 D7 0 0 0 0

   All other steps remain as in CP6 compression.

3. **CP4D compression**

   This is similar to the CP4 mode described above. The only variation is in terms of the way PDMA will compress data chunks.

   A0 B0 C0 D0 A1 B1 C1 D1

   A2 B2 C2 D2 A3 B3 C3 D3

   A4 B4 C4 D4 A5 B5 C5 D5

4. **CP8 compression**

   CP8 compression takes 8 input complex operands and compresses them to one 64-bit words.

5. **CP16 compression**

   CP16 is different from other compression modes. It packs 16 complex operands into 4 64-bit words. It does not use random numbers(LFSR) and does not store common exponents.

6. **CP4FMTA compression**

The CP4FMTA compression involves compressing 4 complex numbers and storing them as a single 32-bit number. Next 4 complex numbers compress to store complete 64 bit number.

In this case the software shall store the numbers as follows

A0 B0 C0 D0 0 0 0 0

A1 B1 C1 D1 0 0 0 0

.....

.....

A7 B7 C7 D7 0 0 0 0

All other steps remain as in CP6 compression.

7. **CP4DFMTA compression**

   This is similar to the CP4FMTA mode described above. The only variation is in terms of the way PDMA will compress data chunks.

   A0 B0 C0 D0 A1 B1 C1 D1

   A2 B2 C2 D2 A3 B3 C3 D3

   A4 B4 C4 D4 A5 B5 C5 D5

8. **CP4FMTB compression**

   The CP4FMTB compression involves compressing 4 complex numbers and storing them as a single 64-bit mantissa and 4 bit exponent.

   There are two different flavours for CP4FMTB. In this case the software shall store the numbers as follows

   A0 B0 C0 D0 0 0 0 0

   A1 B1 C1 D1 0 0 0 0

   .....

   .....

   A7 B7 C7 D7 0 0 0 0

   All other steps remain as in CP6 compression.

9. **CP4DFMTB compression**

This is similar to the CP4FMTB mode described above. The only variation is in terms of the way PDMA will compress data chunks.

A0 B0 C0 D0 A1 B1 C1 D1

A2 B2 C2 D2 A3 B3 C3 D3

A4 B4 C4 D4 A5 B5 C5 D5

10. **CP8FMTB compression**

    The CP8FMTB compression involves compressing 8 complex numbers and storing them as two 64-bit mantissas and 4 bit exponent.

11. **CP16FMTB compression**

    The CP16FMTB compression involves compressing 16 complex numbers and storing them as four 64-bit mantissas and 4 bit exponent.

12. **Decompression**

    This runs in the reverse direction with all missing bits being filled with sign extended bit or zero padding depending on the instruction. This procedure generates 8 complex output values per clock cycle for CP6/CP8 and 4 complex output for CP4/CP4D. CP16 decompression is not valid. For CP4FMTB, CP4DFMTB, CP8FMTB, CP16FMTB, mantissa and exponent values are read from different SRAM location starting from SYSRAM_MEM_START_ADDR and SPT_PDMA_FMTB_EXP_ADDR_STATUS respectively.

NOTE

For CP4FMTB, CP4DFMTB, CP8FMTB and CP16FMTB, first exponent is stored or read from address given by SPT_PDMA_FMTB_EXP_ADDR_STATUS. After this same skip and continuous address scheme will be used for both mantissas and exponent. Exponent base address is calculated as:

- For TRANS_TYPE = 1 :

```
if (DATA_PACKING == CP4FMTB_COMPRESSION)
  VECTOR_LEN_2 = VECTOR_LEC>>3
else
  VECTOR_LEN_2 = VECTOR_LEN>>2
```

- For TRANS_TYPE = 0:

```
  VECTOR_LEN_2 = VECTOR_LEN
```

-
```
JUMP_ADDR = VECTOR_LEN_2/SYSRAM_CONTINOUS_ADDR
REMAINDER_ADDR = VECTOR_LEN_2-JUMP_ADDR*
```

```
SYSRAM_CONTINOUS_ADDR
```

- ```
  if(REMAINDER_ADDR == 0)
    MULTIPLE_FACTOR = JUMP_ADDR
  else
    MULTIPLE_FACTOR = JUMP_ADDR + 1
  ```

- ```
  SPT_PDMA_FMTB_EXP_ADDR_STATUS = SYSRAM_MEM_START_ADDR+
  (MULTIPLE_FACTOR*8) * (SYSRAM_CONTINOUS_ADDRESS
  +SYSRAM_SKIP_ADDRESS)
  ```

45.9.9.6 PDMA instruction

Table 45-55. PDMA instruction fields

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(101000)						TAG_N_BITFLD	SE/COMP_RND	IMA	DATA_PACKING					TRANS_TYPE	SYNC_
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
VECTOR_LEN															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SYSRAM_MEM_START_ADDR[31:16]															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
SYSRAM_MEM_START_ADDR[15:1]															Reserved
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
OPRAM_MEM_START_ADDR[15:0]															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
OPRAM_SKIP_ADDR[11:0]												OPRAM_CONTINUOUS_ADDR[11:8]			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPRAM_CONTINUOUS_ADDR[7:0]								SYSRAM_SKIP_ADDR [11:4]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYSRAM_SKIP_ADDR[3:0]				SYSRAM_CONTINUOUS_ADDR[11:0]											

Table 45-56. PDMA instruction fields description

Bit	Bit Field	Description
127:122	OPCODE	Opcode for PDMA command (101000b)
121	TAG_N_BITFLD	1b: Operation needs to be performed on tagged data.Bit Number 24 used to determine if data is tagged. Used in aggregation mode.
		0b:Operation needs to be performed on non zero data(data is packed in terms of bit fields)
120	SE/COMP_RND	Sign Extension/Compression Rnd bit.This bit is multiplexed.

Table continues on the next page...

Table 45-56. PDMA instruction fields description (continued)

Bit	Bit Field	Description
		In non compression mode : Sign Extension bit. This indicates whether input data should be sign extended or zero padded during packing/unpacking • 0b - Zero padding • 1b - Sign Extension
		In compression mode : Compression Rand bit. This indicates whether LFSR random value should be added or fixed value should be added for compression. • 0b - LFSR random value selected • 1b - Fixed value 'h80 selected
119	IMA	0b : No indirect address
		1b : source and destination address are indirect i.e a work register contains the address of the PDMA source and destination
118:114	DATA_PACKING	Defines the data packing type
		00000b -16-bit complex pack
		00001b - 24-bit complex pack
		00010b - 24-bit real pack
		00011b - 16-bit real pack
		00100b - 48-bit binary pack
		00101b - 16-bit complex swap pack
		00110b - 16-bit mode for clubbing 2 channels data
		00111b - CP4 compression
		01000b - CP6 compression
		01001b - CP8 compression
		01010b - CP16 compression
		01011b - CP4D compression
		01101b - Absolute aggregation
		01110b - Index value aggregation
		01111b - Index Cross Sum aggregation
		10000b - CP4FMTA compression
		10001b- CP4DFMTA compression
		10010b- CP4FMTB compression
		10011b- CP4DFMTB compression
		10100b- CP8FMTB compression
		10101b-CP16FMTB compression
113	TRANS_TYPE	Direction of transfer from PDMA
		0b: System RAM to Operand/Twiddle RAM
		1b : Operand/Twiddle RAM to System RAM
112	SYNC	1b : Synchronous Data transfer. Current PDMA Transfer needs to finish before next instruction is initiated.
		0b : Asynchronous Data Transfer. Current transfer does not need to finish for the next instruction to be initiated.

Table continues on the next page...

Table 45-56. PDMA instruction fields description (continued)

Bit	Bit Field	Description
111:96	VECTOR_LEN	Number of Address locations to be read from source side. Source could be SRAM or OPRAM/Twiddle RAM. Vector length should be programmed in such a way so that it results in multiple of 8 number of locations access on OPRAM/Twiddle RAM. For CP4FMTB, CP4DFMTB, CP8FMTB, CP16FMTB, for SRAM to SPTRAM transfer, this field corresponds to the number of 64 bit mantissa words to be read.
95:65	SYSRAM_MEM_START_ADDR	System RAM Address that determines the start location for the transfer. It should be in multiple of 8.
63:48	OPRAM_MEM_START_ADDR	Operand/Twiddle RAM Address that determines the start location for the transfer. This should be in multiples of 8.
47:36	OPRAM_SKIP_ADDR	Number of words (48*8-bit each) that should be skipped after operand/twiddle RAM continuous address are read or written
35:24	OPRAM_CONTINUOUS_ADDR	Number of words (48*8-bit each) that should be read or written before the jump for operand/twiddle RAM. DMA would start with continuous address
23:12	SYSRAM_SKIP_ADDR	Number of words (64-bit each) that should be skipped after System RAM continuous address are read or written
11:0	SYSRAM_CONTINUOUS_ADDR	Number of words (64-bit each) that should be read or written before the jump. DMA would start with continuous address. For CP4FMTB, CP4DFMTB, CP8FMTB, CP16FMTB, this field values should be programmed in power of 2

45.9.9.6.1 Source and destination continuous/skip Address

The PDMA module supports transfer of data between Operand/Twiddle RAM and System RAM. The source and destination can be controlled by the instruction fields. Both source and destination have an option of specifying continuous and skip address.

These fields control the location and amount of data that will be transfered between the memories. These fields are used in conjunction with the System RAM/Operand RAM Start Address and System RAM/Operand RAM Skip Address.

Continuous address controls the number of data (48 *8 bit in case of operand/twiddle RAM and 64 bits in case of System RAM) that will be read/written before the jump which is specified by skip address.

- If continuous address = 0, the SPT module throws a DMA command error.
- If continuous address = 1 for operand RAM/twiddle RAM it means after START address/SKIP address we need to read/write 8 locations before another skip.
- If continuous address = 1 for System RAM it means after START address/SKIP address we need to read/write 1 location (64 bits) before another skip.
- If skip address = 0 it means continuous transfer.
- If skip address = 1 for operand RAM/twiddle RAM it means after every continuous address we need to skip 8 locations

- If skip address = n for operand RAM/twiddle RAM it means after every continuous address we need to skip n*8 locations.
- If skip address = 1 for System RAM it means after every continuous address we need to skip 1 System RAM location(64 bits each).
- If skip = n for System RAM it means after every continuous address we need to skip n System RAM locations(64 bits each).

The user will have to program both of them to transfer chunk of data from one memory to the other. Assuming base address = 0x8000

For Operand/SPT RAM

- Continuous address = 1, Skip address = 0.
- Final Address : 8000,8001,8002,8003,8004,8005,8006,8007,8008, 8009,800A,800B, 800C,800D,800E,800F
- Continuous address = 1, Skip address = 1.
- Final Address = 8000,8001,8002,8003,8004,8005,8006,8007,8010, 8011,8012,8013,8014,8015,8016,8017
- Continuous address = 2, Skip address = 1.
- Final Address = 8000,8001,8002,8003,8004,8005,8006,8007,8008, 8009,800A, 800B,800C,800D,800E,800F,8018, ,8019,801A,801B,801C,801D,801E,801F
- Continuous address = 2, Skip address = 2.
- Final Address = 8000,8001,8002,8003,8004,8005,8006,8007,8008, 8009,800A, 800B,800C,800D,800E,800F,8020, ,8021,8022,8023,8024,8025,8026,8027

For System RAM

- Continuous address = 1, Skip address = 0: Final Address = 8000,8008,8010,8018.
- Continuous address = 1, Skip address = 1: Final Address = 8000,8010,8020,8030
- Continuous address = 2, Skip address = 1: Final Address = 8000,8008,8018,8020
- Continuous address = 2, Skip address = 2: Final Address = 8000,8008,8020,8028

So if a user specifies Operand/Twiddle RAM Memory start address as 0x100, Continuous address as 2 and skip address as 4, the PDMA will start transfering data starting from 0x100 to 0x110 with 16 words(48 bit each) transfered and then will skip the next 32 words and so on.The next continuous address will start from 0x130 to 0x140.

Similarly if a user specifies System RAM Memory start address as 0x100, Continuous address as 2 and skip address as 4, the PDMA will start transfering data starting from 0x100 to 0x110 with 16 bytes transfered and then will skip the next 32 bytes and so on. The next continuous address will start from 0x130 to 0x140.

.

45.9.9.6.2 Data Aggregation

Some operations deliver sparse vector/matrix results and only certain values are interesting for the following processing. For this purpose, it is possible not to transfer the complete vector to System RAM, but only a selected part, which depends on the values itself. The values must meet some criteria, either they must be a max or greater 0

Data Aggregation is only supported for transfer from SPT/Operand RAM to System RAM. This is not supported for transfers from System RAM to SPT/Operand RAM.

The following modes are supported for data aggregation:

- Absolute, only values which are Max or greater 0 are transferred
- Indexed value, for any value which is a Max or greater 0 its offset to the source address (element 0) and the value itself is transferred, all other values are skipped
- Indexed cross sum, for any value which is a Max or greater 0 its offset to the source address (element 0) and the sum of all bits in the value is transferred, all other values are skipped

The aggregation mode works with only 16 bit values.To explain aggregation with an example, let us consider the following data that needs to be transferred to system RAM:

0: i10939 + 12876

1: i63846 + 0

2: i0 + 56617

3: i43918 + 28847

4: i0 + 54068

5: i23263 + 0

6 :i57885 + 0

7: i0 + 0

When transferring using aggregation mode for value only, following is the format:

0:12876,10939,63486,56617

1:28847,43918,54068,23262

2:57885,....,.....,....

All values greater than zero are transferred.

When transferring using aggregation mode value and index, following gets transferred:

0: 0,12876,1,10939

1: 3,63846,4,56617

2: 6,28847,7,43918

3: 8,54068,11,23263

4: 13,57885,.....

In the above example all values real and imaginary which are > 0 are transferred along with index with base index starting from element 0 (Real part of first number)

Similarly when transferring data using aggregation mode indexed cross sum, following gets transferred:

0: 0, 6,1, 9

1: 3, 10, 4, 9

2: 6, 9, 7, 9

3: 8, 8, 11, 11

Here along with the index from base value, sum of all the bits is transferred for the numbers which are greater than zero. So for number 10939 (10101010111011) we transfer 9 as the bit sum.

45.9.10 DMA Arbiter

45.9.10.1 DMA Arbitration

The SPT module has just one AHB master interface with 3 different sources of data i.e SDMA, PDMA & Command sequencer queue DMA. Each of these modules provides the DMA arbitration logic with the interface details in terms of the number of data that needs to be transferred, address of the transfer and whether the module intends to read or write data from/to System RAM. It should be noted that SDMA has two sub-sources, the main SDMA and the bypass SDMA, and these are arbitrated inside the Acquisition module.

PDMA and command data transfers can be both stalled, however the amount of data of PDMA transfers is significantly higher than for commands. It is important to note that since ADCs provide continuous stream of data, the transfers from the acquisition/SDMA cannot be delayed.

The default priority scheme assumes the following order:

- Acquisition main SDMA or bypass SDMA write (top priority)

- RAM transfer PDMA
- Command DMA read (lowest priority)

All DMA transfers generate bursts of data. Arbitration between different sources can only take place at the borders of bursts. The Bursts will never be broken. PDMA gives control on the number of data that needs to be transferred via the instruction format. However this control is not available for SDMA & CSDMA. Tables below show the various burst sizes that are being used for main SDMA, since for transfers from main SDMA these are controlled by the different modes supported in main SDMA. For bypass SDMA always a single burst of INCR16 is initiated.

Table 45-57. Burst Size in main SDMA for various acquisition Modes/Channels enabled when source is ADC or Complex mode from MIPICSI2

MODES	Interleaved	Tile 4	Tile 8
NUM_CHNL = 1	16 (INCR16)	2(2XINCR)	2(2XINCR)
NUM_CHNL = 2	16 (INCR16)	4 (INCR4)	4 (INCR4)
NUM_CHNL = 3	16 (INCR16)	6(INCR4+2XINCR)	6(INCR4+2XINCR)
NUM_CHNL = 4	16 (INCR16)	8(INCR8)	8(INCR8)
NUM_CHNL = 5	16 (INCR16)	10(INCR8+2XINCR)	10(INCR8+2XINCR)
NUM_CHNL = 6	16 (INCR16)	12(INCR8+INCR4)	12(INCR8+INCR4)
NUM_CHNL = 7	16 (INCR16)	14(INCR8+INCR4+2XINCR)	14(INCR8+INCR4+2XINCR)
NUM_CHNL = 8	16 (INCR16)	16(INCR16)	16(INCR16)

Table 45-58. Burst Size in main SDMA for various acquisition Modes/Channels enabled when source is MIPICSI2

MODES	Interleaved	Tile 4	Tile 8
NUM_CHNL = 1	16 (INCR16)	4(INCR4)	4(INCR4)
NUM_CHNL = 2	16 (INCR16)	8(INCR8)	8(INCR8)
NUM_CHNL = 3	16 (INCR16)	12(INCR8+INCR4)	12(INCR8+INCR4)
NUM_CHNL = 4	16 (INCR16)	16(INCR16)	16(INCR16)

The PDMA however behaves differently than the SDMA. The burst size is not fixed. Instead based on the number of data that needs to be transfered, the DMA arbiter initiates the transaction on the AHB bus. PDMA has an output FIFO which continues to fill till there is 16 data in the FIFO. This is the default setting in case the amount of data to be transferred is large. The arbiter then initiates an INCR16 on the AHB bus to transfer the data. In case the amount of data is less, then the PDMA pre-calculates the number of data that needs to be transferred and accordingly adjusts the burst size.It is however possible to control the burst that PDMA initiates based on setting in the control registers. The software can ensure SPT only initiates INCR4/INCR8 or INCR16 only.

It is also possible to control the size of each burst that is initiated on the AHB bus by Command Sequencer by programming field SPT_CS_MODE_CTRL[CSDMA_BRST_SZ].

Table 45-59. AHB Master ID for SDMA, PDMA and Command Sequencer

Source	AHB Master ID
Main SDMA	14
Bypass SDMA	10
PDMA	6
Command Sequencer	13

Back to back transactions on the PDMA will have no idle cycles. However when there is a switch between PDMA/SDMA/Command sequencer, one cycle will be lost in arbitrating and initiating a new burst.

In order to provide the Command sequencer access within a long PDMA burst so that it does not keep waiting until PDMA finishes, the SPT supports a 16 bit register SPT_PDMA_CONTROL[PDMA_COUNT_CS_PRIORITY] which needs to be programmed. This specifies the number of clock cycles after which the command sequencer will get access. The PDMA will finish the existing burst and handover the access to the command sequencer.

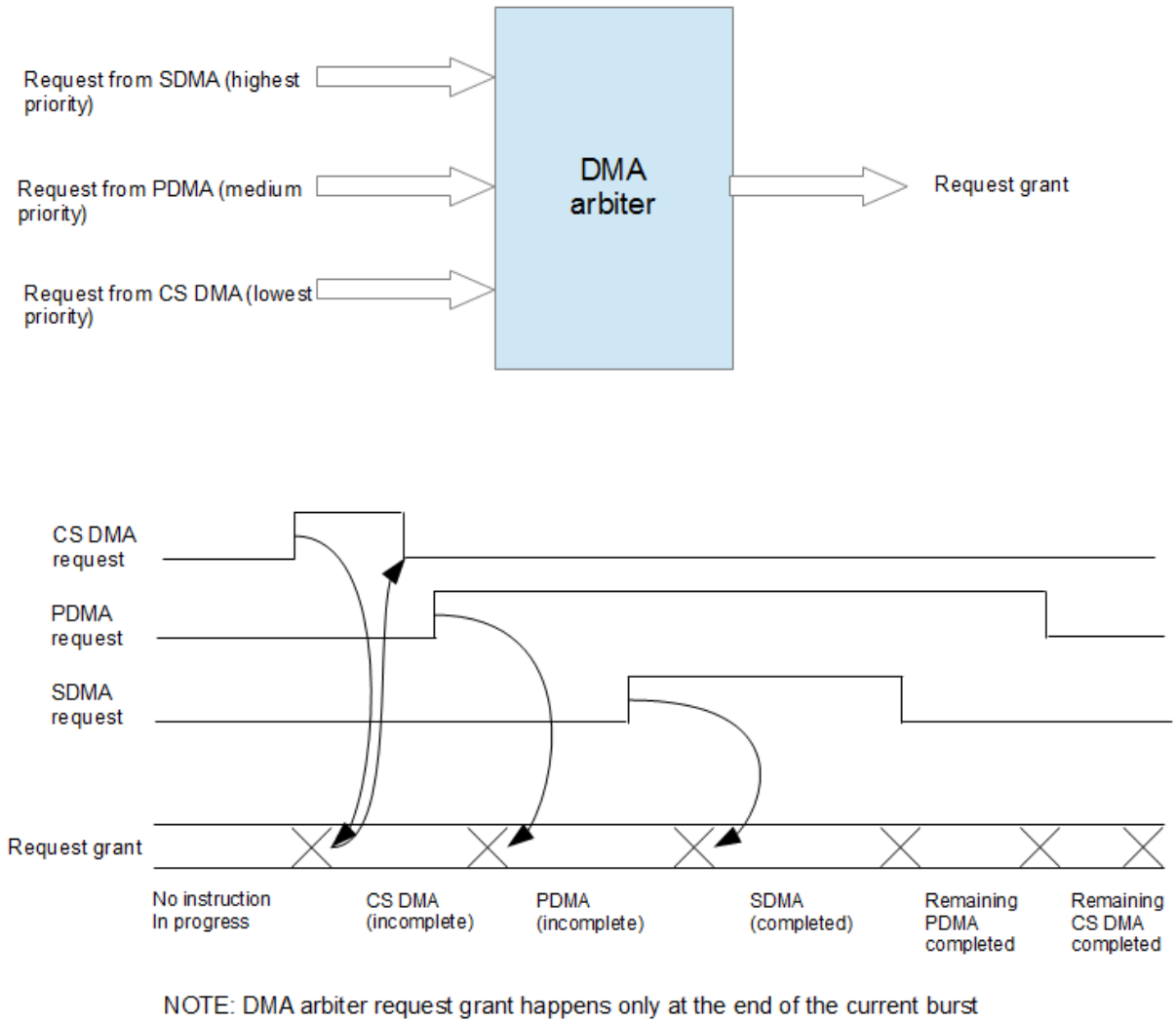


Figure 45-59. DMA arbiter

45.9.11 VMT

45.9.11.1 Introduction

Vector Math is a collection of basic operations on vector data. The operations are performed on 8 operands in parallel for the given length of vector. Some operations can be concatenated within a single processing step.

45.9.11.2 Principle of operation

The vector math unit is implemented as fully pipelined structure with several stages performing certain mathematical operations. Flow diagram shows the 2 operations sequences and pipelining in each operation(shown as dotted lines). Each dotted line corresponds to one pipeline cycle in the design and to be accounted for throughput analysis.

In Operation Sequence 2, the very first stage performs either Abs or Mag or Conjugate operation and forwards its results to the next stage, which performs a Shift or Offset or Sum with second vector operation. The final stage takes this results and sums up the vector elements to each element. Each operation stage can also be bypassed.

In Operation Sequence 1, square of Magnitude is followed by logarithm 2.

Regardless of the source of input operands being Operand/Twiddle RAM, the input operand is sign extended to 24 bits and saturated. Saturation consists of clamping this value of input operand between 0x80_0001 and 0x7F_FFFF. The pipeline width is fixed to 24 bits from here onwards. No intermediate saturation stages exist except for Operation sequence 2, stage 2 operations; where result is saturated to 24 bits. Finally, output data is clipped to 16 bits (MSB's discarded) if destination is Twiddle RAM. This followed by a 24b/16b saturation logic based on destination RAM selected thus arriving at the result.

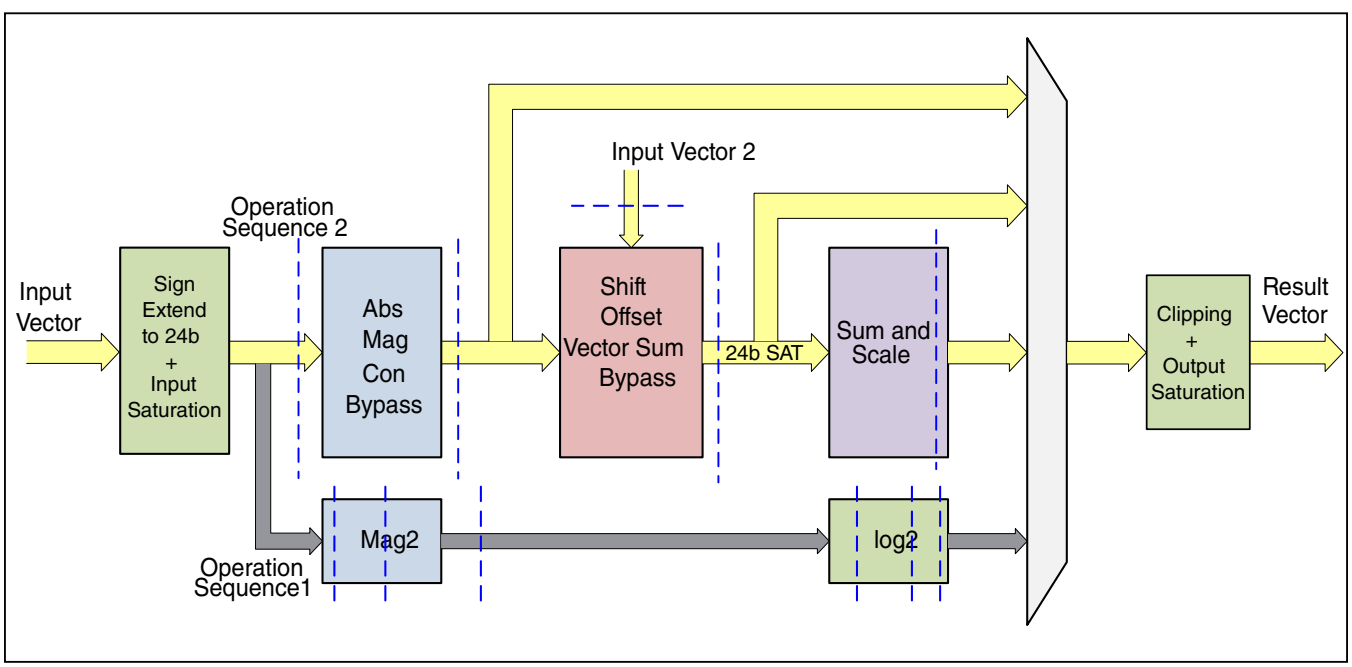


Figure 45-60. Vector Math Operation flow diagram

45.9.11.3 Vector Math Operations

The following 2 sequences of operations are supported:

45.9.11.3.1 Operation Sequence 1

The first sequence of the Valid VMT Operations as defined by field OP_SEQ1 consists of 2 stages as shown in lower branch of the VMT flow diagram

45.9.11.3.1.1 Stage 1 - Vector Mag2 operation

This operation has to be performed when Operation sequence 1 is selected. From the original 24-bit operands, approximated 12-bit operands are chosen based on the location of their leading '1' bit. The operation calculates the sum of the square magnitude value of each approximated operand.

The resulting sum has 25-bit resolution, and is given to the sub-module performing the log2 operation. The other output of the mag2 module is the exponent value corresponding to the location of the leading '1' bit.

This operation is supported for real and complex operands only. The output values are real numbers.

45.9.11.3.1.2 Stage 2 - Vector Log2 Operation

It converts the input vector to log2 numbers. The operation makes use of the binary number representation:

$$log_2(n) = log_2\left(\sum_{k=0}^{MSB} a_k 2^k\right)$$
$$= log_2\left(2^{MSB} \sum_{k=0}^{MSB} a_k 2^{k-MSB}\right)$$
$$= MSB + log_2\left(\sum_{k=0}^{MSB} a_k 2^{k-MSB}\right)$$

Figure 45-61. Formula for Log2 Calculation

The 6 leading bits (exp) are derived by counting in the log2 module's input the position of the first MSB bit that is ‘1’, where the LSB is assigned the position 0 (From this exp is subtracted the intermediate exponent value coming from the mag2 module, and a correction is done by adding 22. This gives the final exponent). The 8 LSB bits following this leading bit are used for mantissa calculation.

The log2 output occupies only 14-bit and is always a positive number. An input value of 0 results in log2 output as 0.

45.9.11.3.2 Operation Sequence 2

The second sequence of the Valid VMT Operations consists of 3 stages as shown in the VMT flow diagram.

NOTE

In this operation sequence, all 3 stages cannot be bypassed.
Atleast one math operation must be selected.

45.9.11.3.2.1 Stage 1:

45.9.11.3.2.1.1 Vector Abs

It calculates the absolute value of each operand and is selected when OP_SEQ2[5:4] = 01. It is supported for real and complex operands. The output values are real numbers.

Real operands:

$y_k = |x_k|$, $k = 0..N-1$

Complex operands :

$y_k = (|\ Re(x_k)\ | + |\ Im(x_k)\ |) >> 1,\ k = 0..N-1$

The absolute value is limited to 24 bit and has only positive numbers.

45.9.11.3.2.1.2 Vector Conjugate

It calculates the conjugate complex value of each operand. It is supported complex operands only. This is the only preprocessing operation which has complex output values.

$y_k = Re(x_k) - j\ Im(x_k)$, $k = 0..N-1$

45.9.11.3.2.1.3 Vector Magnitude

Operation calculates half of approximate root of square magnitude value of each operand. It is supported for complex operands only. The output values are real numbers.

45.9.11.3.2.2 Stage 2:

45.9.11.3.2.2.1 Vector Shift

This operation shifts the input operands by the number of positions (absolute value of b) to the left (positive b) or right (negative b). It is supported for real and complex operands. The input operands are sign extended for right shift and LSBs are lost. With left shifting ,'0' LSB bits are inserted. The value b is read from the WORK REGISTER whose address is specified by VMT instruction fields [15:0]. The bit positions [5:0] of selected WORK REGISTER location provide the value b. The valid value for this field is of 6 bits where 6th bit is sign(which decide on right or left shift) and the absolute value of this 6 bit signed field gives the actual shift value. The absolute shift value (b) is saturated to 24 for values above 24.

If a left shift operation results in overflow (sign change) for any operand, then the result is saturated to 24 bits, i.e Negative overflow causes result to saturate to 24'h80_0001, whereas positive overflow causes result to saturate to 24'h7F_FFFF. Moreover the overflow status bit HW_ACC_ERR_STATUS[VMT_SHFT_OVF_ERR] is also set on detecting overflow occurrence for any operand undergoing this operation.

Real operands:

$y_k = x_k << b,\ k = 0..N\text{-}1$

Complex operands :

$y_k = Re(x_k) << b + j\ Im(x_k) << b\ ,\ k = 0..N\text{-}1$

45.9.11.3.2.2.2 Dual Vector Sum

This operand sums each element of first vector with each element of second vector. It is supported for real and complex operands. If input type is complex, stage 1 (pre-processing) operation cannot be Vector Abs or Vector Magnitude. First Vector Source is Operand RAM and second vector is located in the Twiddle RAM. The address increment for second vector source is fixed to 1. The result is saturated to 24 bit signed values and can only be stored in Operand RAM.

$y_k = x1_k + x2_k,\ k = 0..N\text{-}1$

45.9.11.3.2.2.3 Vector Offset

It adds an offset (constant value) to each vector element Supported for real and complex operands. The value C can be provided as an immediate value or can be read from the Work Register in which case work register fields [15:0] provide $Re(C_k)$ whereas fields [39:24] provide $Im(C_k)$. The result is saturated to 24 bit signed values and can be stored in Operand/Twiddle RAM.

$y_k = (Re(x_k) + Re(C_k)) + j\ (Im(x_k) + Im(C_k))\ ,\ k = 0..N\text{-}1$

45.9.11.3.2.3 Stage 3:

45.9.11.3.2.3.1 Vector Sum and Scale

This operation is valid for real operands only. It calculates the sum of all vector elements and generates a single output value. The result can be scaled with a multiplier value S and a divider value D, which is applied last. S is a signed 16bit number. The divider value D must be a power of two; i.e. 2 raised to the power N, where N is the exponent. The values S and N are read from the Work Register where the field [15:0] provides value of S while N is given by the field [29:24] of the selected Work Register Location. N is a 6 bit value as shown in the figure below. The 48 bit signed result can be stored in operand RAM or Work Register.

$$y = \frac{S}{D}\sum_{k=0}^{N-1} x_k$$

The internal accumulator bit width is 53 bits. It is possible to start the accumulation from scratch or to continue with a previous accumulated value based on RST_ACC field in VMT instruction set. The value of exponent N must be chosen appropriately to ensure that final result can be stored within 48 bits. In the worst case, upto 5 MSB's from 53 bit result will get discarded to trim down the result to 48 bits.

45.9.11.4 Input and Output Packing

Input Packing is selected by the IN_PACK field in VMT instruction. This field is valid only when the input data type (IN_DATTYP) is real. Input operands can either be packed, unpacked real or unpacked imaginary. When input is packed, each slice of the Operand/Twiddle RAM can hold 2 operands.

Output Packing is selected using OP_PACK field in VMT instruction. When output is unpacked, the decision of whether output is unpacked real or unpacked imaginary is taken based on IN_PACK value. Programming Input to be unpacked real leads to output being unpacked real. Similarly if input is unpacked imaginary, output would be unpacked imaginary as well.

The following table provides the relationship between input data type, packing options and their effects on various other parameters.

Table 45-60. Input and Output Packing

	Possible Scenarios									
	1	2	3	4	5	6	7	8	9	10
Input Data Type	real	real	real	real	real	real	complex	complex	complex	complex
Input Packing	real (00)	real (00)	imag (01)	imag (01)	packed (10)	packed (10)	x	x	x	x
Output Packing	off	on	off	on	off	on	off	off	on	x
Vector Length	n	n	n	n	n	n	n	n	n	n
VMT result data type	real	real	real	real	real	real	real	real	real	complex
No. of accessed input addresses	n	n	n	n	n/2	n/2	n	n	n	n
No. of accessed output addresses	n	n/2	n	n/2	n	n/2	n	1	n/2	n
Output real part	vmt result re	vmt result re1	keep	vmt result re1	vmt result re	vmt result re1	vmt result re	sumscale result [23:0]	vmt result re1	vmt result re
Output imag part	keep	vmt result re2	vmt result re	vmt result re2	keep	vmt result re2	keep	sumscale result [47:24]	vmt result re2	vmt result im
Comments								sumscale output only		

45.9.11.5 VMT Operations - Source and Destination Options

When performing a Vector Math operation, it is mandatory to program the instruction fields to select any of the supported source and destination address locations. The following figure shows all valid source and destination options for various VMT math operations.

		Valid Source	Valid Destination
Sequence 1	MAG2	OPRAM, TWRAM	OPRAM, TWRAM
	LOG2		
Sequence 2, Stage 1	ABS	OPRAM, TWRAM	OPRAM, TWRAM
	MAG	OPRAM, TWRAM	OPRAM, TWRAM
	CONJ	OPRAM, TWRAM	OPRAM, TWRAM
Sequence 2, Stage 2	SHIFT	OPRAM, TWRAM	OPRAM, TWRAM
	OFFSET	OPRAM, TWRAM	OPRAM, TWRAM
	SUM	OPRAM, TWRAM *	OPRAM
Sequence 2, Stage 3	SUM & SCALE	OPRAM, TWRAM	OPRAM, WREG

*For Dual Vector Sum operation, first vector is picked from OPRAM and second vector is picked from TWRAM simultaneously.

Figure 45-62. Valid Memory Options for VMT Operations

45.9.11.6 VMT Instruction Format

The table below shows the instruction bit encoding used for VMT. The details of each bit field and their operation is defined in the sections below.

Table 45-61. VMT Instruction format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100110)						RST_ ACC	IN_DA TTYP	OP_S Q1	OP_SQ2						OP_P ACK
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
IP_PACK		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
MULT_S_EXP_N_VAL_ADDR															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table 45-61. VMT Instruction format (continued)

IMDT_OFFSET_VAL_IM[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2ND_VEC_SRC / SHIFT_VAL / OFFSET_VAL_ADDR / IMDT_OFFSET_VAL_RE[15:0]															

Table 45-62. VMT Instruction Fields Description

Bit	Bit Field	Description
[121]	RST_ACC	Reset Sum Scale Accumulator
		0 - Does not reset sum-scale accumulator. If current instruction executes sum-scale operation, new sum is accumulated over previous accumulated sum.
		1 - Resets sum-scale accumulator on start of instruction.
[120]	IN_DATTYP	Input Data type
		0 - Real
		1 - Complex
[119]	OP_SQ1	Operation Sequence 1
		1 - Log 2 + Mag2
		0 - Sequence 2 selected
[118:117]	OP_SQ2 - Stage1	Operation Sequence 2 - Stage1
		00 - No Preproccessing
		01 - ABS (for real and complex numbers)
		10 - Magnitude for complex values (calculate half of approximate square root of real^2 + im^2 - only valid for complex numbers)
		11 - Conjugate(Valid for complex numbers)
[116:114]	OP_SQ2 - Stage2	Operation Sequence 2 - Stage2
		000 - No Operation
		001 - Shift
		010 - Offset with Value in Work Register
		011 - Offset with Immediate value
		100 - Dual Vector sum(Only be used when Vector Source is Operand RAM and 2nd Vector Source from Twiddle RAM)
[113]	OP_SQ2 - Stage3	Operation Sequence 2 - Stage3
		0 - No Operation
		1 - Sum and Scale
[112]	OP_PACK	Output Packing - This field allows user to select types of Output Data Packing.
		0 - Output unpacked. 24 bit re/im used depending on IN_PACK selected. If IN_PACK = 01 (imaginary unpacked), then 24 bit im is used; otherwise 24 bit re is used.
		1 - Output packed (48 bit used)
[111:110]	IN_PACK	Input Packing - This field allows user to select types of Input Data Packing.
		00 - Input unpacked real (24 bit real used)

Table continues on the next page...

Table 45-62. VMT Instruction Fields Description (continued)

Bit	Bit Field	Description
		01 - Input unpacked imaginary (24 bit im used)
		10 - Input packed (48 bit used)
[109]	IMA	Indirect Memory Addressing
		0 - Immediate Memory Addressing
		1 - Indirect Memory Addressing for Source and Destination addresses
[108:96]	VEC_SZ	Vector size/length. The value of this field must be even when input packing is selected (IN_PACK = 10b).
[95:80]	SRC_ADD	Source Address (should be multiple of 8)
[79:64]	DEST_ADD	Destination Address(should be multiple of 8 except for sum and scale operation i.e. when OP_SQ2 - Stage3 = 1)
[63:48]	MULT_S_EXP_N_VAL_ADDR	Address of Working Register where multiplier (S) and exponent (N) value are stored. From of the 48 bit word stored in this location, bits [15:0] give value of multiplier (S), while bits [29:24] give value of exponent (N).
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[31:16]	IMDT_OFFSET_VAL_IM[31:16]	Imaginary part of immediate offset value used when OP_SQ2[116:114] = 011.
[15:0]	2ND_VEC_SRC_ADDR	As Second vector source address which is required for Dual Vector Sum selected by OP_SQ2[116:114]=100 (should be multiple of 8)
	SHIFT_VAL_ADDR(b)	As Address of Shift Value B if OP_SQ2 - Stage2 = 001
	OFFSET_VAL_ADDR	As Address of Offset Value C when OP_SQ2 - Stage2 = 010
	IMDT_OFFSET_VAL_RE[15:0]	Real part of immediate offset value if OP_SQ2 - Stage2 = 011

45.9.12 HIST

45.9.12.1 Introduction

The HIST module performs a histogram computation operation which counts the number of values falling in certain bins. 16/32/64 bins are supported. Histograms are calculated with 8 or 16 operands parallel over a vector of size N. A histogram over a vector of input data is calculated by scheduling multiple histogram kernel runs until all vector elements are processed.

45.9.12.2 Input data type

Histograms can be generated from vectors with real or complex or log2 operands. The real and complex numbers are either 16-bit or 24-bit wide. In case of log2 operands, the operation shall support also packed operands, i.e. two 14-bit lsb-aligned log2 values within 48bit. This is optional, so a command parameter is provided to decide whether log2 is packed or not. Thus, for log2 data type, 8 or 16 operands may be received with each memory read cycle (but only a single histogram will be calculated for the entire data).

Similar to the log2 data-type case, there is the option for packing 2 16-bit or 24-bit real operands in one 48-bit location. One option in the unpacked log2 or unpacked real cases is that the data input can be accepted either from the real part or the imaginary part.

The input data comes from the operand RAM only.

45.9.12.3 Result storage

The state of the histogram counters can be saved in operand RAM, twiddle RAM or work registers.

45.9.12.4 Operation modes

The Histogram operation supports several operation modes:

- Read operands, calculate histogram and save result to operand RAM, Twiddle RAM or work registers. In this option the bin counters are reset at the beginning of the command.
- Read operands, calculate histogram. In this option the bin counters are reset at the beginning of the command.
- Read operands, calculate cumulated histogram. In this option the bin counters are NOT reset at the beginning of the command.
- Save results to operand RAM, Twiddle RAM or work registers. In this option the bin counters are NOT reset at the beginning of the command.

45.9.12.5 Kernel architecture

The histogram module consists of the preprocessing operation, logarithmic coding operation, the 'compare and count' operation and the histogram bin counters. The preprocessing and the log coding operations are bypassed in case the operands are already log2 values.

45.9.12.5.1 Preprocessing

The preprocessing module enables the 'histogram processing' of different input number formats. If necessary, signed input values are converted to positive values. The following options are available:

- Abs for real input values, negative values are inverted
- Abs for complex input values, as sum of absolute of real and imaginary part
- Mag for complex values, calculates half of approximate square root of the sum of $real^2$ and im^2

The pre-processing is bypassed for log2 numbers.

For real data it is also possible to have the option of no preprocessing.

45.9.12.5.2 Log Coder

The basic algorithm is to count the bit position of the leading '1' from the MSB in each operand and append the required number of bits following the first ‘1’ MSB in order to get the index of the bin. All numbers are treated as unsigned numbers for the purpose of log coding.

The number of appended bits depends on the word length of operand and the desired number of bins. With 16-bit input data the position code has a length of 4 bits. This number can be used directly with 16 bins and 1 or 2 mantissa bits must be appended to get 32 or 64 bins respectively.

With 24-bit operands the position code has a length of 5 bits. This number can be used directly with 32 bins or 1 mantissa bit must be appended to get 64 bins.

If the input data is of log2 type then by default bin size of 64 is chosen.

Table 45-63 covers all the possible cases:

Table 45-63. Valid log coding operations

Input data type	Data size	Bin size	Appended mantissa bits
Real or complex	16-bit	16	0
Real or complex	16-bit	32	1
Real or complex	16-bit	64	2
Real or complex	24-bit	32	0
Real or complex	24-bit	64	1
Log2	Not applicable	64 (default)	Not applicable

45.9.12.5.3 Compare and Count

This module performs a comparison of the log coded number and puts equal numbers in the same bin.

The command word needs to specify a low threshold bin. All values below or equal to this threshold are counted in the specified bin. It must be ensured that the threshold value specified in the command word is not greater than the bin size, else no bins are filled.

45.9.12.5.4 Bin counter

The bin counters are incremented by the count of operands belonging to same bin. The bin counters are cleared with the start of a histogram operation. To generate cumulative histograms, the clearing can be suppressed. The limit for the number of samples for single histograms is defined by the size of the bin counter. The bin counters are protected from overflow and provide an overflow flag in the configuration map. When any counter reaches its overflow value it is saturated at that value (0xFFFF). The result of histogram operation can be saved to operand RAM, twiddle RAM or work registers with the same command or a special command.

The size of every bin counter is 16 bits.

45.9.12.5.5 Writing bin counter values to memory:

Bin counter values can be written to operand RAM, twiddle RAM and work registers. While writing to operand or twiddle RAM, 16 bins can be written in one cycle (8 values of data x2 for real and imaginary parts). While writing to work registers all the bin counter values can be written to the work register in 1 cycle; two bins can be accommodated in one work register, 1 in the real part and 1 in the imaginary part.

Data is written out to operand RAM or twiddle RAM as shown in Figure 45-63. (example for 64 bins)

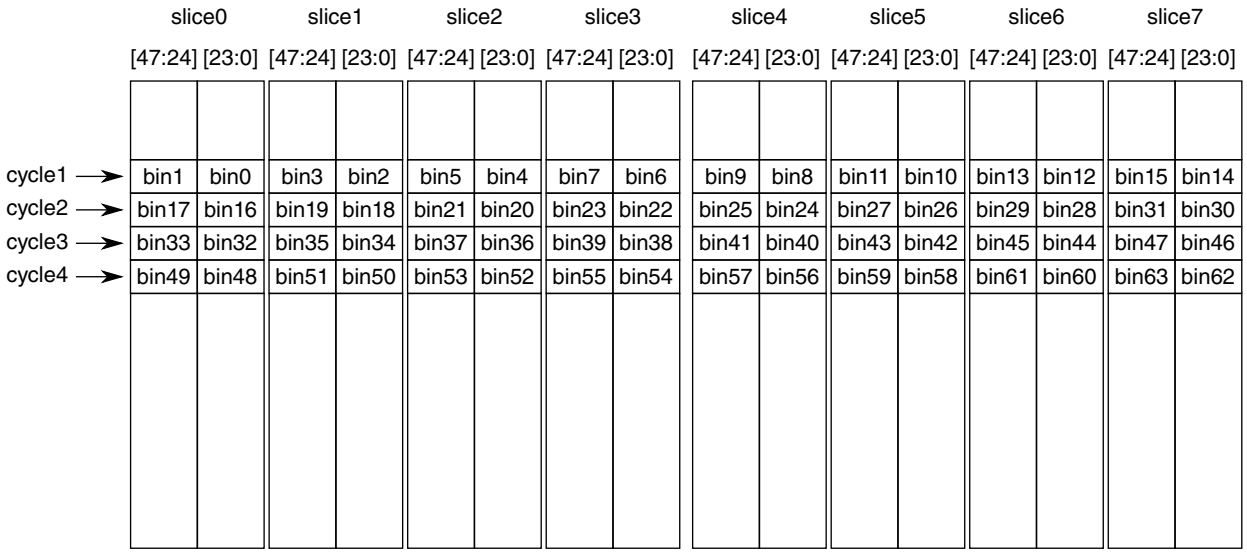


Figure 45-63. Writing bins back to memory

Data is written out to work registers as shown in Figure 45-64.

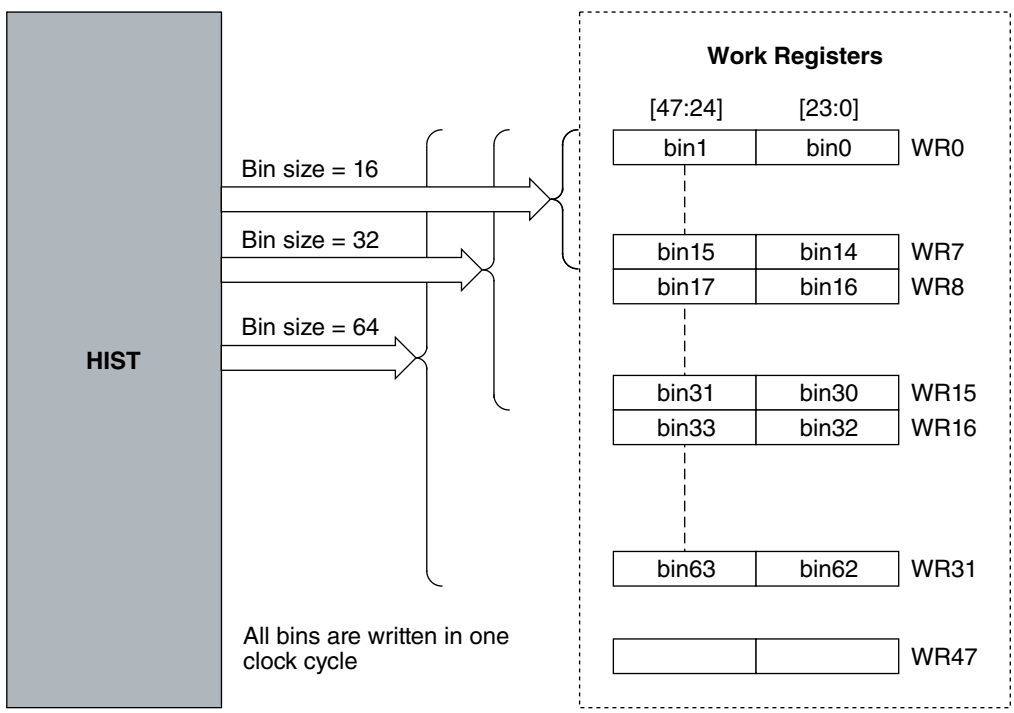


Figure 45-64. Writing bins back to work registers

45.9.12.6 HIST instruction format

The command word for the HIST instruction is shown in Table 45-64

Table 45-64. HIST Instruction Format

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100100)						DAT_TYP		PREPROC		OP_MOD		BIN_SZ		Reserved	
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
PACK_EN		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THR						Reserved									DAT_ SZ

The various command parameters for the HIST instruction are explained in Table 45-65

Table 45-65. HIST Instruction bit description

Bits	Bit Field	Description
[121:120]	DAT_TYP	Input data type
		00 - real
		01 - complex
		1X - log2
[119:118]	PREPROC	Preprocessing (this field is checked only for real and complex input data)
		00 - for real input data type it corresponds to 'no preprocessing', for complex input data type it corresponds to abs operation
		01 - abs for real and complex input data type
		1X - for real input data type it corresponds to abs operation, for complex input data type it corresponds to mag operation

Table continues on the next page...

Table 45-65. HIST Instruction bit description (continued)

[117:116]	OP_MOD	00-read operand RAM, calculate hist and store results. In this option the bin counters are reset at the beginning of the command.
		01-read operand RAM, calculate histogram. In this option the bin counters are reset at the beginning of the command.
		10-read operand RAM, calculate cumulated histogram. In this option the bin counters are NOT reset at the beginning of the command.
		11-store results. In this option the bin counters are NOT reset at the beginning of the command.
[115:114]	BIN_SZ	00-for 16-bit input data it corresponds to 16 bins, for 24-bit input data it corresponds to 32 bins
		01-32 bins
		1X-64 bins
[111:110]	PACK_EN	2-bit field, enables packing of log2 or real input operands. For complex operands this field is ignored. The LSB decides whether the input value comes from the real component or the imaginary component
		00-Packing not enabled for log2 or real operand and data comes from real component
		01-Packing not enabled for log2 or real operand and data comes from imaginary component
		1X-Packing enabled for log2 or real operand.
[109]	IMA	Decides whether addressing is immediate (address comes from the instruction itself) or indirect.
		0-Immediate addressing
		1-Indirect addressing; the address comes from a work register for the source as well as the destination. This work register index is the bits 8:3 of the SRC_ADD field of the command word. The work register index should be from 0 to 47 else an error condition results.
[108:96]	VEC_SZ	Vector length- the vector length is the number of input data and so for the case of packed log2 or real input data it should be twice the number of storage

Table continues on the next page...

Table 45-65. HIST Instruction bit description (continued)

		locations. It is always a multiple of 8, else an error condition is raised. This field is ignored when OP_MOD = 11b
[95:80]	SRC_ADD	Source operand address. This address should be of the operand RAM only and should be a multiple of 8, else an error condition results. This field is ignored when OP_MOD = 11b
[79:64]	DEST_ADD	Destination operand address. Destination can be operand RAM, twiddle RAM or work registers. In case of work registers only the top 2-bits of the address are read and the histogram bin counters are stored in work register number 0 to work register number 31. If the address is of the operand or twiddle RAM then the address should be a multiple of 8, else it is an error condition. This field is ignored when OP_MOD = 01b or 10b
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment. This increment value is valid only when the destination is operand RAM or twiddle RAM.
[15:10]	THR	Minimum threshold bin.
[0]	DAT_SZ	Size of real and complex input data
		0 - 16-bit data
		1 - 24-bit data

45.9.13 MAX-S

45.9.13.1 Introduction

This operation searches for local or global maxima in a vector of data. A global maxima is the single value of a vector exceeding the value of each other vector element. This value can be present multiple times in a vector but it is only found once and the operation delivers a single result. A local maxima is defined as a value that is larger than both its neighbors in a vector.

The search for maxima is one of the most time-consuming operations with RADAR signal processing, therefore a highly-effective HW acceleration is necessary.

45.9.13.2 Principle of operation

The operation can find maxima in a vector of elements. To find the global maximum, the result of the current maximum operation is also compared with the result of the previous maximum operation and overwrites this result, if it is larger. To find the local maximum (peak), each value is compared with its nearest neighboring values. Optionally, the vector is padded with one more value at the beginning and the end and the search range is extended to include also the element 0 and N-1 in the search range.

A two dimentional local maxima search is implemented by multiple SPT command sequences. First, local maxima are determined in each column vector and output is tagged. After this step, the vectors are transposed and the local maxima search is repeated with each column vector again, this time with tagged inputs. The output of the second local maxima operation is in the form of bitfields. There can be an additional PDMA step which will output the aggregated list.

For local or absolute maxima calculations, comparisons are made using ">=" operator.

The MAXS module supports +ve as well as –ve input numbers. In case the incoming numbers are –ve (PREPROC = 00), and a –ve number comparison is needed, the IN_TAG and TAG_N_BITFIELD fields in the instruction need to be set to 0. Therefore, in effect, -ve number comparison can only be done when the input is not tagged and the output is not expected to be tagged.

45.9.13.2.1 Local/Absolute Maxima Definition

A global maximum is defined in the following way:

$$\text{if } (x_n >= x_k),\ gmax = x_n, \text{ where } k = 0...N-1$$

The local maximum of a vector is defined as all values x that meet following equation

$$\text{if } (x_{n-1} < x_n >= x_{n+1}),\ lmax_k = x_n, \text{ where } n = 0...N-1$$

The output of the local maxima operation can be a tagged output or a set of bitfields with a '1' on all the locations with a local maxima. The output of absolute or global maxima is the global maxima value. In case of absolute maxima, the final maxima value that is written is a pre-processed value and not the original 48 bit input.

LOC_N_ABS bitfield of the instruction defines if a local or absolute maxima is to be calculated. A '1' in this bitfield defines that a local maxima has to be calculated while a '0' on that bit implies a global maxima calculation.

45.9.13.3 MAXS Instruction Format

The table below shows the instruction bit encoding used for MAXS. The details of each bit field and their operation is defined in the sections below.

Table 45-66. MAXS Instruction

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
OPCODE(100111)						IN_DATTYP		PREPROC		THLD_CMP	IN_TAG	LOC_N_ABS	TAG_N_BITFLD	CYC_EXTN	MAXSN_EN
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
IN_PACK		IMA	VEC_SZ												
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
SRC_ADD															
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
DEST_ADD															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reserved															
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
SRC_ADD_INC								DEST_ADD_INC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														MAXSN_SEL	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THLD_ADD															

Table 45-67. MAXS Instruction bit description

Bits	Bit Field	Description
[121:120]	IN_DATTYP	Input Data Type
		00 - Real
		01 - Complex
		10 - Log2
		11 - Reserved
[119:118]	PREPROC	Pre-processing
		00 - No preprocessing (for log2 and others)
		01 - ABS(real) + ABS(im) (for real and complex numbers. Imaginary part ignored in case of real input)
		10 - Magnitude for complex values (calculate half of approximate square root of real^2 + im^2 - only valid for complex numbers)
		11 - Reserved
[117]	THLD_CMP	Threshold Compare (valid only for local maxima)

Table continues on the next page...

Table 45-67. MAXS Instruction bit description (continued)

Bits	Bit Field	Description
		0 - Threshold compare disabled
		1 - Threshold compare enabled
[116]	IN_TAG	Input tagged (Only valid for log2 datatype as input and for local maximum calculation)
		0 - Input is not tagged
		1 - Input is tagged
[115]	LOC_N_ABS	Local not Global maxima
		0 - Global maxima to be calculated
		1 - Local maxima to be calculated
[114]	TAG_N_BITFLD	Tag not Bitfield (valid only for local maximum calculation)
		0 - Output to be packed bitfields
		1 - Output to be tagged vectors
[113]	CYC_EXTN	Cyclic extension (valid only for local maximum calculation)
		0 - No cyclic extension
		1 - Cyclic extension
[112]	MAXSN_EN	MAXSN enable
		0 - MAXSN disabled
		1 - MAXS to be calculated for multiple blocks of N operands at one time
[111:110]	IN_PACK	Input packing
		00 - Input unpacked real (24 bit real used)
		01 - Input unpacked imaginary (24 bit im used)
		10 - Input packed (48 bit used)
[109]	IMA	Indirect Memory Addressing
		0 - Immediate Memory Addressing
		1 - Indirect Memory Addressing for Source and Destination addresses
[108:96]	VEC_SZ	Vector size/length
[95:80]	SRC_ADD	Source Address
[79:64]	DEST_ADD	Destination Address
[47:40]	SRC_ADD_INC	Source address increment
[39:32]	DEST_ADD_INC	Destination address increment
[17:16]	MAXSN_SEL	MAXSN operand Multiplicity select
		00 - MAXS16
		01 - MAXS8
		10 - MAXS4
		11 - reserved
[15:0]	THLD_ADD	Threshold address

NOTE

Destination Increment is ignored for global max and local max bitfield output

45.9.13.3.1 Input Tagged

The input to the MAXS operation may be tagged or untagged. In case the input is in tagged format, the local maxima search only checks the tagged locations with its neighbours. If the tagged location is greater than its neighbours, it is declared a local maxima. The tagged inputs are required for the software enabled two dimentional local maxima search.

In case of global maxima, the tag bit is ignored and is removed while saving the global maxima.

The MSB (24th bit) bit is used as tag bits. Only operand of datatype log2 can be tagged. Tagging is not allowed for real or complex inputs datatypes.

45.9.13.3.2 Input Datatype

The maxs operation is possible with input data in the real, complex and log2 formats. Two bits in the instruction format are used to define the type of input to the MAXS operation.

The Table 45-68 describes the different pre-processing requirements with different types of input formats. Two bits in the instruction format are used to define the type of complex pre-processing required.

Table 45-68. Input Datatype

Input Data	PreProcessing required	Number of bits
Real	No preprocessing	24
	Abs(real) + Abs(Im)	
Complex	Abs(real) + Abs(im)	24
	Mag for complex values (calculate approx sq root of real^2 + im^2)	
Log	No Preprocessing	14

45.9.13.3.3 Input Packing

Max search can work on up to 16 operands in parallel. For this purpose, the input data need to be available in packed format, so that with 8 inputs from operand RAM, 16 24-bit operands can be processed concurrently.

In case of local maxima operation with packed input operands, if TAG_N_BITFLD bit of the instruction is set, both the 23rd and the 47th bit of the output are tagged for local maximas. Complex input datatype are always preprocessed, hence is never in packed format.

Bitfield IN_PACK of the instruction defines if the input is coming in packed format. A '10' on this bitfield implies that the input is in packed format. A '0' on the MSB signifies an unpacked format or 1 operand per 48 bits of incoming data.

45.9.13.3.4 Threshold Compare

When Threshold Compare is enabled by setting THLD_CMP bitfield of the instruction, any local maxima value is compared with the threshold value as well. And a maxima is reported only when the value is greater than the threshold. Threshold compare option is only valid for local maxima. When global maxima is selected, the THLD_CMP bitfield of the instruction is ignored. The threshold value is a second input operand for a max operation. It can be located in work registers or twiddle RAM. No preprocessing is done on the threshold value and it is used as is. In case of TAG_N_BITFIELD or IN_TAG set to 1, the threshold value should be positive, that is, the 16^{th} bit in case of read from TRAM and 24^{th} bit in case of read from OPRAM/Work registers should be 0.

It is advisable to store the threshold values in one of the work registers. Storing threshold values in OPRAM or TRAM leads to initial latency of single cycle. In addition if source of operands and threshold values are same, it leads to lower performance.

45.9.13.3.5 Output Tag/Bitfield

Local maxima output is generated based on the TAG_N_BITFLD field of the instruction. When TAG_N_BITFLD is set to zero, the local maxima outputs are generated as a string of bits with 1's in locations having a local maxima. The bits are packed in chunks of 16 and written to the LSBs of the 24-bit word in case of OPRAM and work registers. The rest of the 8 bits are set to 0. In case of twiddle RAM all 32 bits are occupied.

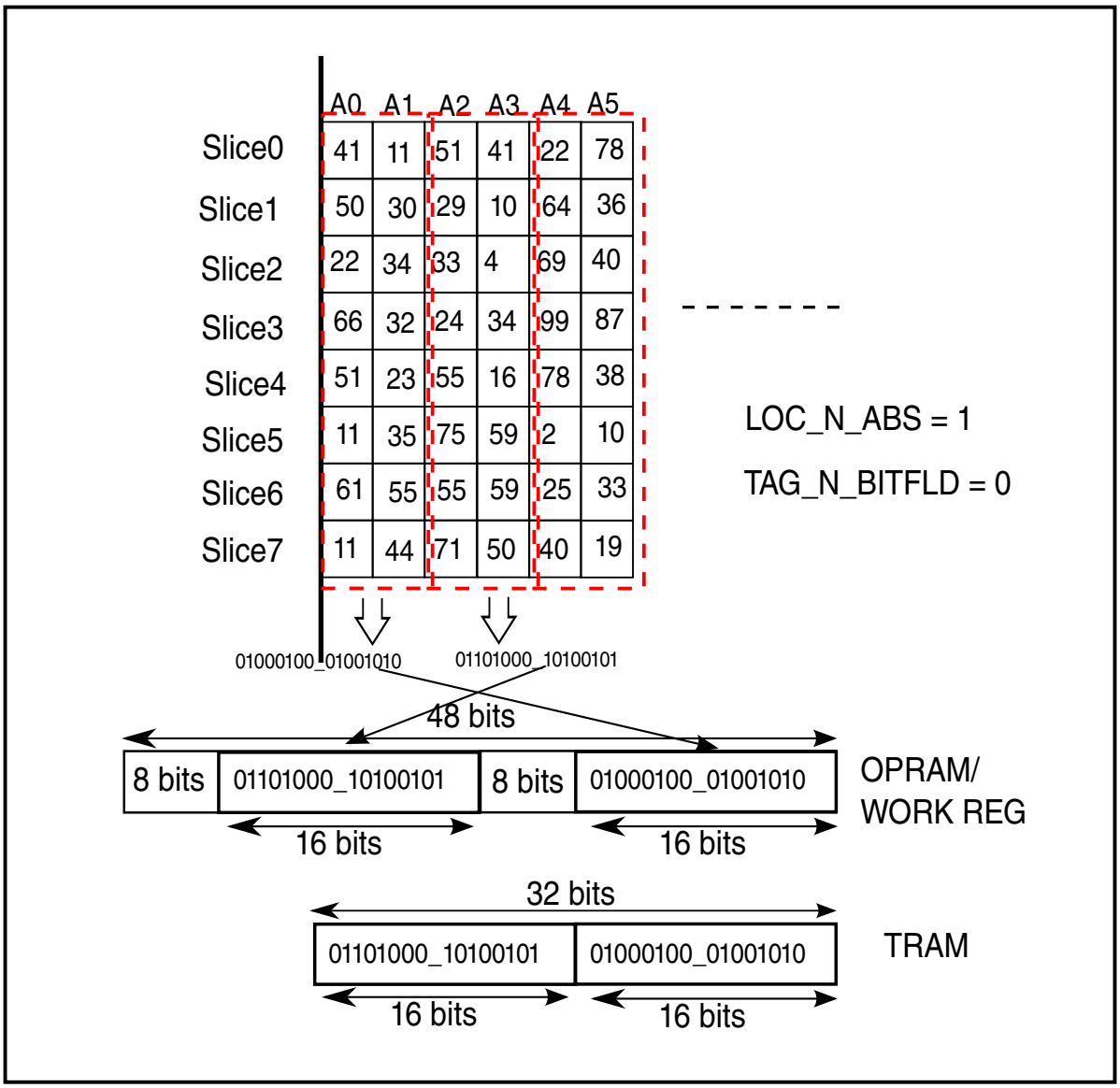


Figure 45-65. Local max bitfield packing

When TAG_N_BITFLD option is set to 1, the output vectors are simply tagged and processed inputs (not original inputs). The 24th bit is used as the tag field. The input vectors are processed, tagged and written to the DEST_ADD. The SRC_ADD remains unaltered.

In case of packed operands (IN_PACK = 10b), the bitfield for the operand occupying the Real location will go to the LSB.

The destination for tagged outputs can only be OPRAM as the 24th or 48th bit of the input word is set to 1. While the bitfields are always written in a packed manner, the tagged outputs in case of local max are packed differently. The following table shows how the outputs are packed.

Table 45-69. MAXS Output packing

loc_n_abs	in_dattype	Output Real	Output Im
Local max (output tag)	COMPLEX	Tagged output	Unchanged
	REAL/LOG (in_pack = 00b)	Tagged output	Unchanged
	REAL/LOG (in_pack = 01b)	Unchanged	Tagged output
	REAL/LOG (in_pack = 10b)	Tagged output	Tagged output

45.9.13.3.6 Cyclic extension

Cyclic extension allows the detection of local maximum at the very first or the very last vector element. After cyclic extension, the first element of the vector is also compared with the very last element of the vector and vice-versa. CYC_EXTN field in the MAXS instruction defines whether cyclic extension is to be used in the maxima calculation. Cyclic extension is only applicable with local maxima search.

45.9.13.3.7 MAXS N

MAXS N is a special operation in which multiple blocks of N operands are processed at one time. The number of operands in one operation is always equal to N, where N can be 4,8,16 given by MAXSN_SEL in the instruction. The total number of blocks is determined as VEC_SZ divided by N. The basic motivation of having this option is that the latency between the various MAXN operations are reduced. Figure 45-66 shows an example of absolute maxima MAXSN for N=16 operation executed for 16 blocks. This operation generates 16 maxima values.

All local and absolute maxima description apply for MAXSN as well, the only difference being that the operation will be carried out for multiple blocks.

As the number of generated results in case of MAXSN can be large, the destination address should be chosen carefully. WREG can only be used as destination for MAXS16. It cannot be used for MAXS8 and MAXS4 as we get multiple global maxima outputs in each cycle for these.

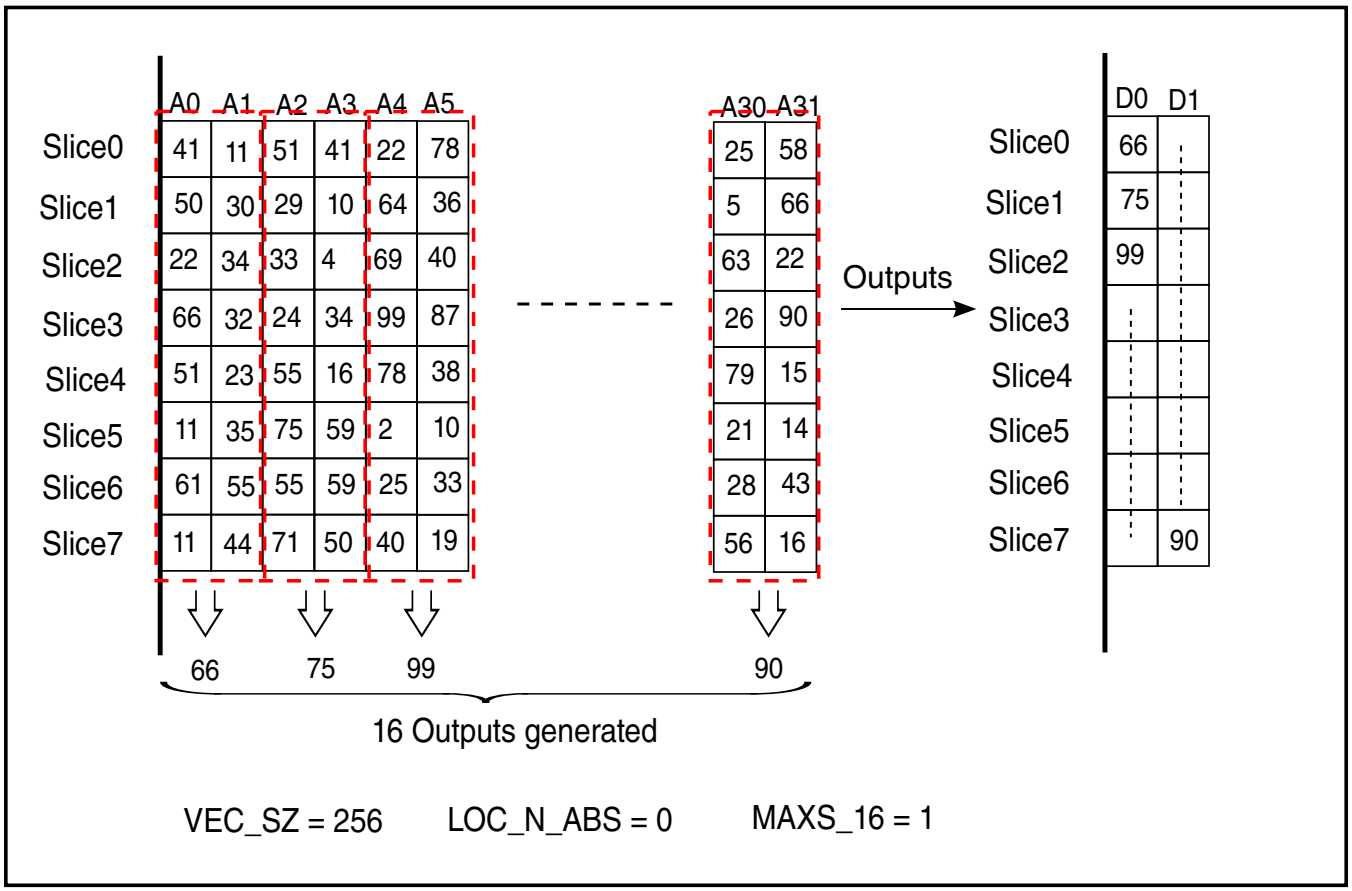


Figure 45-66. MAXS 16 for 16 blocks

When THLD_CMP is enabled (valid for local maximum only), a new threshold value is read for every new MAXSN block. For MAXS4 packed format, 4 threshold values have to be read at once, so the threshold address should be a multiple of 4. Similarly, for MAXS4 unpacked format or MAXS8 packed format, 2 threshold values have to read at once, making the address a multiple of 2. The threshold value is a preprocessed value which is read from the Real part of the vector. For MAXSN, the threshold value can only reside in TRAM and the input data can not reside in TRAM.

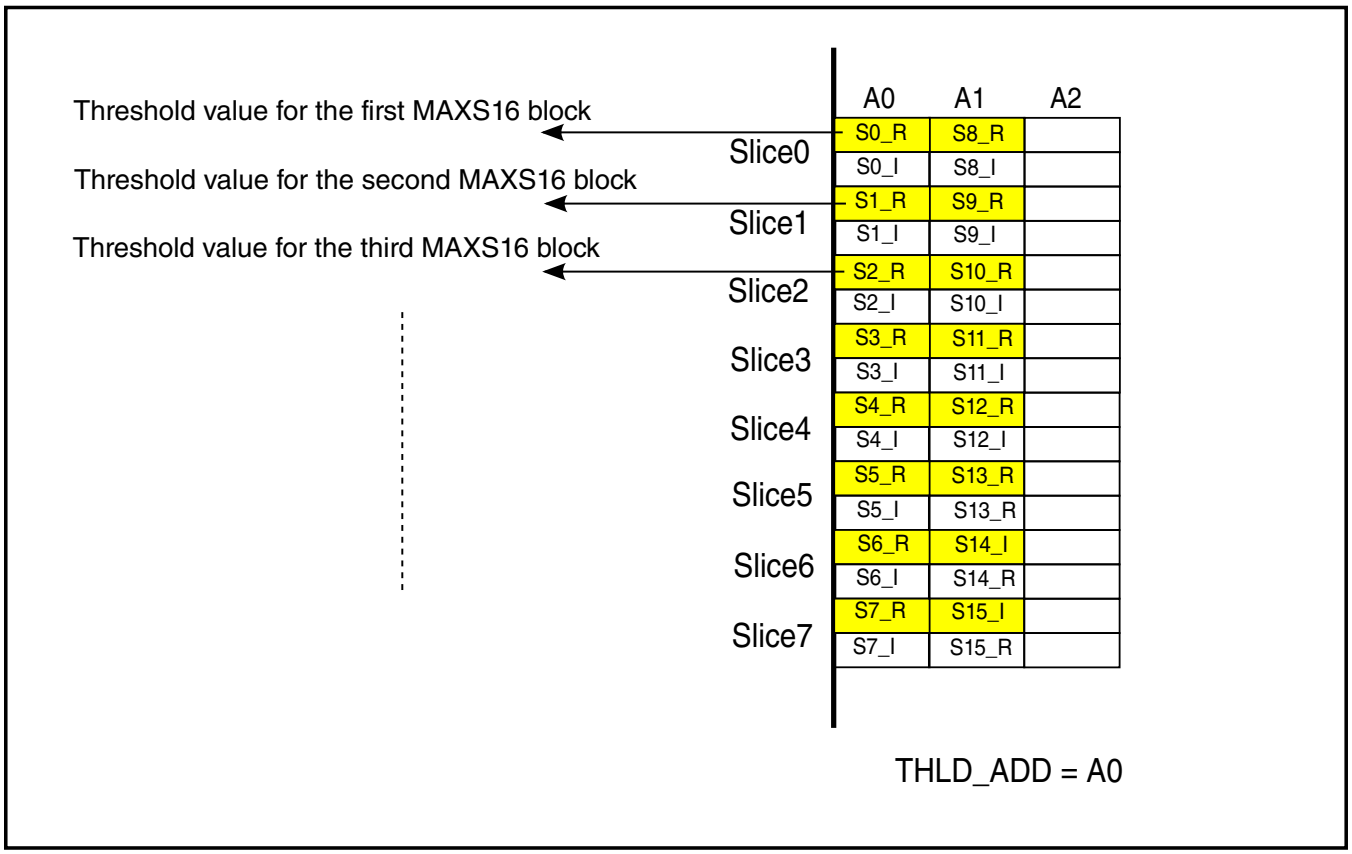


Figure 45-67. Threshold for MAXS16

45.9.13.4 Error scenarios

Maxs gives out a command error if the instruction is not programmed correctly. Table 45-70 gives the list of possible cases.

Table 45-70. Error Scenarios

ERROR	DESCRIPTION
tagged_data_log_err	local maxima with in_tag and datatype is not LOG2
tagged_data_src_add_err	in_tag data for local maxima and source address is not OPRAM
tagged_output_dest_add_err	Tagged local maxima output and destination is not OPRAM
maxsn_thld_add_err	MAXSN threshold compare for local maxima and threshold address is not TRAM
maxsn_src_add_err	MAXSN threshold compare for local maxima and source address is not OPRAM
maxsn_dest_add_err	MAXSN threshold compare for local maxima and destination address is TRAM
cmplx_data_pckd_err	data type is complex and in_pack set to 2'b10 (packed)
cmplx_no_preproc_err	data type is complex and preproc set to 0 (No preproc)

Table continues on the next page...

Table 45-70. Error Scenarios (continued)

ERROR	DESCRIPTION
dattype_rsvd	input data type = 2'b11
in_pack_rsvd	input packing = 2'b11
preproc_rsvd	preproc = 2'b11
maxsn_rsvd	maxsn_sel = 2'b11
maxs16_vec_sz_err	maxsn_sel = 2'b0 (maxs16) and vector size not a multiple of 16
vec_sz_err	vector size not a multiple of 8
packed_vec_sz_err	in_pack = 2'b10 (packed) and vector size not a mulitple of 16
src_add_wreg_err	source address is WREG
dest_add_wreg_err	local maxima and destination is WREG
maxs8_4_dest_wreg_err	maxs8 or maxs4 enabled and destination is WREG. Maxs16 supports WREG.
maxs8_4_thld_addr_err	maxs8 packed/maxs4 unpack, threshold address not multiple of 2. Maxs4 packed, threshold address not multiple of 4.

45.10 Interrupts

The SPT can schedule interrupts to the CPU. The interrupts are defined with event commands and specific flags, which are visible to the CPUs as interrupt flags. Interrupt flags can be masked out from interrupt generation. The CPU can also poll to specific interrupt flags. To offload CPU from polling, special connections are provided directly to Cores and they will become part of the Version Register in CPU. Dedicated Connections to Core1 are from the status bits of SPT_CORE1_VER_EVT and to Core2 from the status bits of SPT_CORE2_VER_EVT. The order is also same as in the status register fields. These connections are shown in Figure 45-68

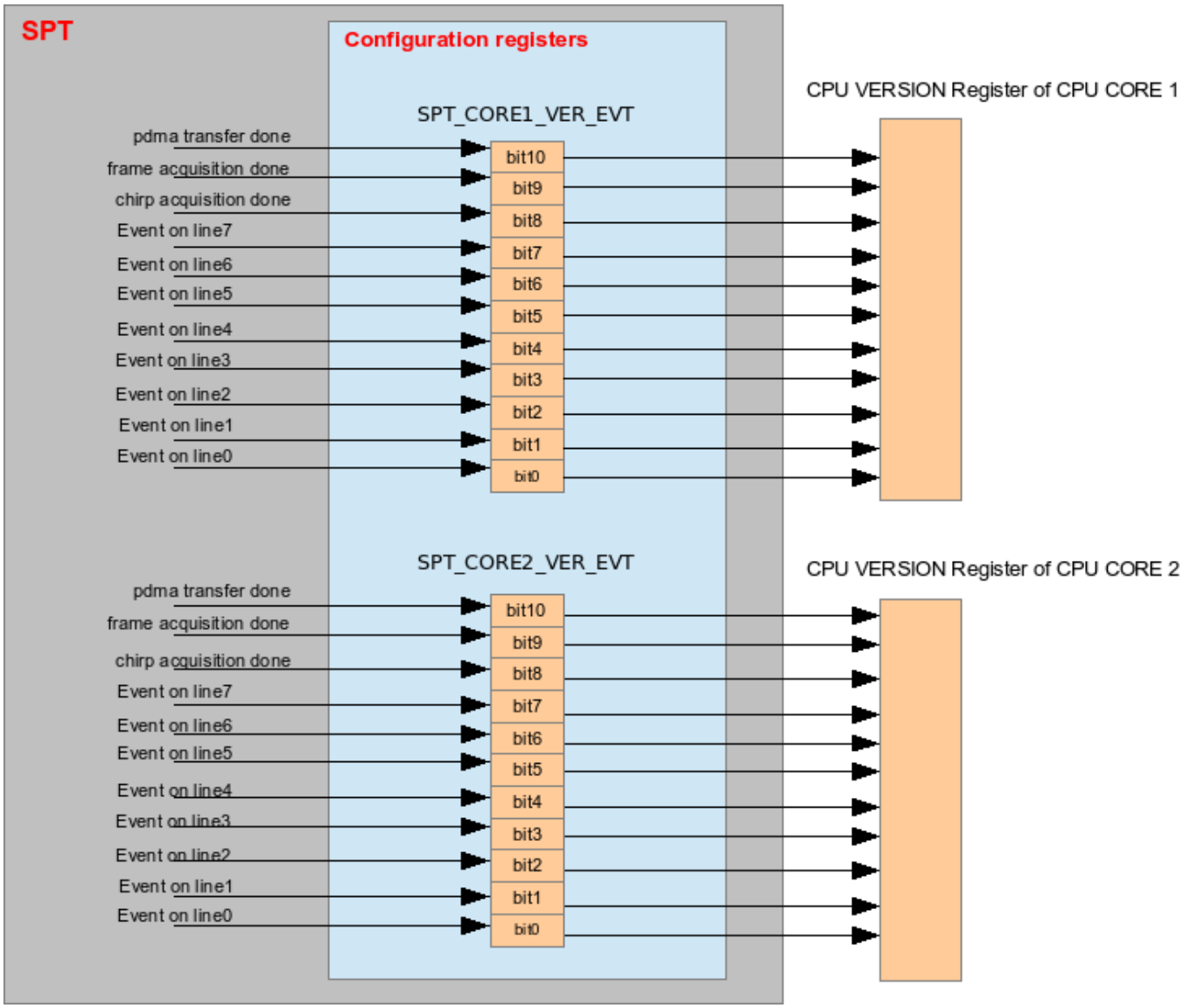


Figure 45-68. Connections between SPT and CPU VERSION Registers

Special Interrupts are connected to all the Cores and Cores can select themselves whether to watch them or ignore them. It is assumed that either a single core handles these interrupts or each core responds only a specific interrupt flag and resets it. If more than one core is using the interrupt events, then individual interrupt handler of each core will be invoked on every interrupt event, to check the state.

Following are the interrupt lines to INTC:

- SPT_IRQ_ECS - It is the general interrupt line which includes the command sequencer related status interrupts as well as the all error and overflow interrupts. SPT_IRQ_ECS consists of the following interrupt and interrupt enable register pairs:

  SPT_MEM_ERR_STATUS - SPT_MEM_ERR_INT_EN

SPT_DMA_ERR_STATUS - SPT_DMA_ERR_INT_EN

SPT_HW_ACC_ERR_STATUS - SPT_HW_ACC_ERR_IE

SPT_HIST_OVF_STATUS0 & SPT_HIST_OVF_STATUS1 - SPT_HIST_OVF_IE

SPT_CS_STATUS0 - SPT_CS_INTEN0

SPT_CS_STATUS1 - SPT_CS_INTEN1

- SPT_IRQ_DMA - This interrupt line includes the PDMA Transfer Done signal, Chirp Acquisition Done and Frame Acquisition Done signals. SPT_IRQ_DMA consists of the following interrupt and interrupt enable register pair:

  SPT_GBL_STATUS - SPT_GBL_STATUS_IE

- SPT_IRQ_EVT1 - This line includes the SPT event interrupts. This interrupt is for the CPU CORE1. SPT_IRQ_EVT1 consists of the following interrupt and interrupt enable register pair:

  SPT_CS_EVTREG1 - SPT_CS_EVT1_INTEN

- SPT_IRQ_EVT2 - This line includes the SPT event interrupts. This interrupt is for the CPU CORE2. SPT_IRQ_EVT2 consists of the following interrupt and interrupt enable register pair:

  SPT_CS_EVTREG2 - SPT_CS_EVT2_INTEN

The interrupt signals along with the WAIT and event generation capability provide hooks to the CPU to synchronize its operations with the SPT operations and with the incoming ADC/MIPICSI2 data frequency. The SPT can wait for 8 software events generated on the peripheral interface, 6 events coming from CTE (or in case of MIPICSI2 the hsync and vsync events), the PDMA transfer done signal, Chirp Acquisition Done and Frame Acquisition Done signals. It also has the capability to program the type of awaited event i.e. whether the awaited event is positive/negative edge triggered or positive/negative level triggered.

The EVT command can throw out events which can be captured by CPU on the peripheral interface.

Figure 45-69 shows the interrupt, WAIT and EVT strategy of the SPT.

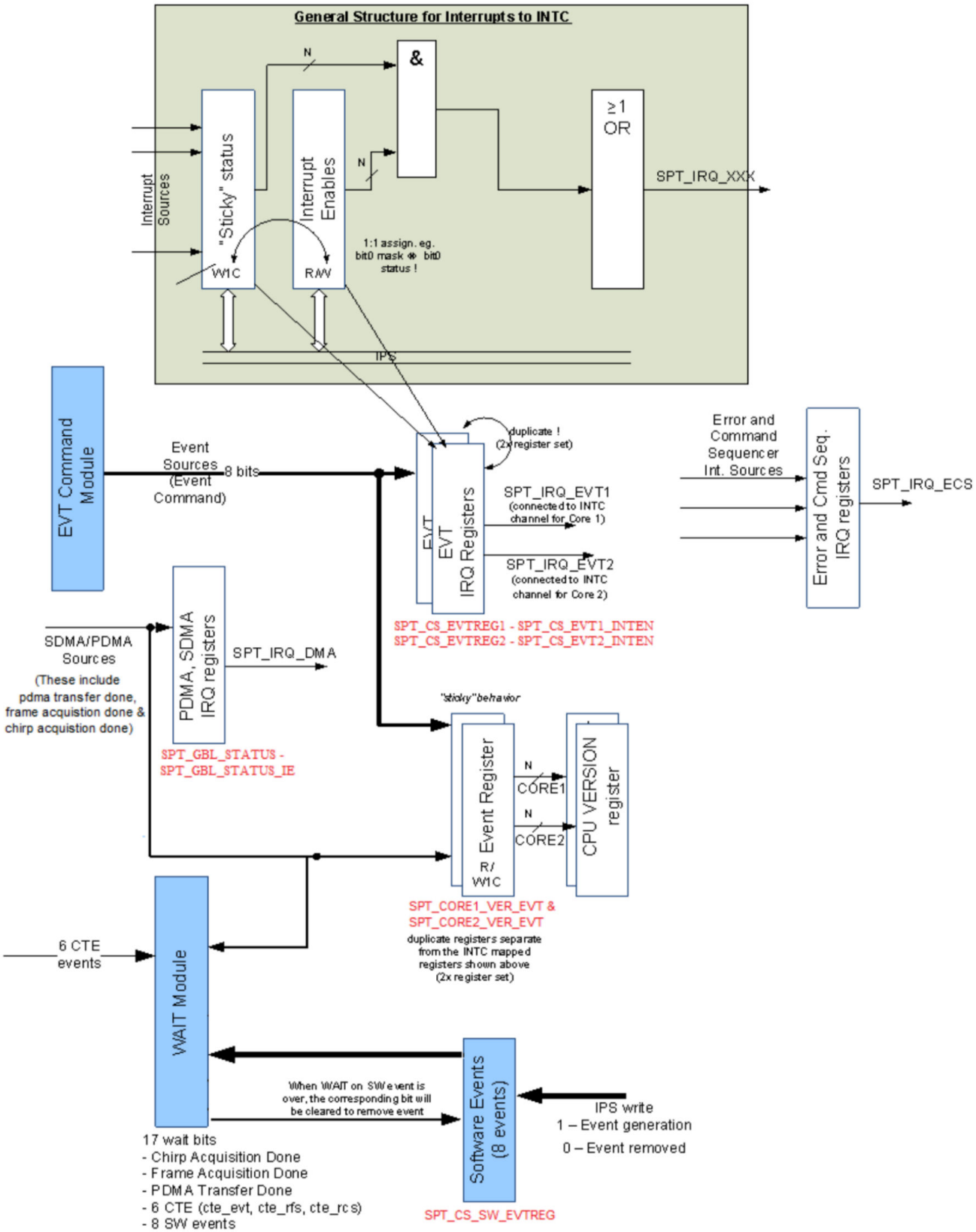


Figure 45-69. SPT interrupts, WAIT and EVT strategy

45.11 Initialization Sequence

The following steps give the sequence for SPT initialization:

1. The user must ensure that the instruction sequence is loaded into the SRAM or Flash or any Slave memory.
2. Program the Acquisition and SDMA Configuration registers. Configure the following configuration registers as per requirement: SPT_ACQ_GBL_CTRL_0, SPT_ACQ_GBL_CTRL_1, SPT_ACQ_CTRL0, SPT_ACQ_CTRL1, SPT_ACQ_CTRL2, SPT_ACQ_CTRL3, SPT_SDMA_CTRL0, SPT_SDMA_CTRL1. In case of MIPI mode, also configure SPT_ACQ_CSI_CTRL, SPT_SDMA_BYP_CTRL0, SPT_SDMA_BYP_CTRL1, SPT_ACQ_BYP_CTRL.
3. Then enable SPT_GBL_CTRL[ACQ_EN] so that Acquisition of ADC sample can start after trigger from CTE is received or input samples from MIPICSI2 can start after vsync from MIPICSI2 is received. User can choose not to enable ACQ_EN when usecase is to process data from the memory only.
4. Update the SPT_CS_MODE_CTRL register to decide the working mode for the command sequencer. The command sequencer can be initialized in the RUN state or in the DEBUG state. To initialize the sequencer in the DEBUG state the SPT_CS_MODE_CTRL[CS_IMM_DEBUG] or SPT_CS_MODE_CTRL[CS_SYS_DEBUG] field needs to be set. By programming the SPT_CS_MODE_CTRL[DEBUG_MD] the user can set the debugging mode to step once mode, step jump mode, instruction jamming mode or halt mode. User can also set upto 4 breakpoints in the debug mode by programming SPT_CS_MODE_CTRL[BKPTx_EN], SPT_CS_MODE_CTRL[BKPTx_RE] and SPT_CS_BKPT0_ADDR. SPT_CS_MODE_CTRL[PREFETCH] field can be set to enable the early pre-fetch option. Also update SPT_CS_PG_ST_ADDR which gives the location from where the Command Sequencer starts fetching data upon start-up.
5. Program SPT_CS_MODE_CTRL[CSDMA_BRST_SZ] as per requirement. This field should be held constant when the program is running.
6. Configure SPT_PDMA_LFSR_LOAD_VAL_HIGH, SPT_PDMA_LFSR_LOAD_VAL_LOW and SPT_PDMA_CTRL registers appropriately to decide the configuration for PDMA usage and priority.
7. User may also set the appropriate bits of the various interrupt enable registers.
8. Enable SPT_GBL_CTRL[SPT_EN] to enable the SPT module. Then after 10 SPT clock cycles set the SPT_GBL_CTRL[PG_ST_CTRL] field which enables Command Sequencer to fetch first 16 instructions from the address programmed in SPT_CS_PG_ST_ADDR. At this point, it is advisable for the CORE to wait for the

sequencer to go to RUN state (or DEBUG state) before writing to the peripheral interface. After this all the hardware acceleration modules will process as per instructions decoded in the sequence.

45.12 Throughput

Table 45-71 shows the number of cycles taken for implementing the various control commands:

Table 45-71. Table showing the number of cycles for control commands' implementation

Command	Mode	Total number of clock cycles
SET	Data source is immediate data	2
	Data source is Work Register	3
GET	All modes	3
ADD	All modes	5
WAIT	-	1(and then command sequencer goes to WAIT state)
EVT	-	1
WATCHDOG	-	1
LOOP	-	1
NEXT	-	1
SUB	All modes	5
CMP	All modes	5
SEL	-	5
JUMP	-	2+number of cycles to re-fill the instruction queue

Table 45-72 shows the additional pipelining cycles and also the total number of cycles taken from the implementation of the various operation commands.

Table 45-72. Table showing pipelining cycles and total number of cycles for operation commands' implementation

Command	Mode	No. of additional pipeline cycles	Total no. of cycles (throughput)
HIST (reading memory data + processing)	complex data input (abs pre-processing)	6	(Vec_len/8) + 6 + (write_cycles)
	complex data input (mag pre-processing)	7	(Vec_len/8) + 7 + (write_cycles)
	unpacked real data input	6	(Vec_len/8) + 6 + (write_cycles)

Table continues on the next page...

Table 45-72. Table showing pipelining cycles and total number of cycles for operation commands' implementation (continued)

Command	Mode	No. of additional pipeline cycles	Total no. of cycles (throughput)
	packed real data input(but the 16 received data values take 2 cycles to be processed, so there is no throughput gain)	6	(Vec_len/8) + 6 + (write_cycles)
	unpacked log2 data input	5	(Vec_len/8) + 5 + (write_cycles)
	packed log2 data input (in this case all the 16 data inputs are processed in one clock cycle, so double throughput)	5	(Vec_len/16) + 5 + (write_cycles)
HIST (write_cycles)	Destination is work register	1	-
	16 bins and destination is TRAM or OPRAM	1	
	32 bins and destination is TRAM or OPRAM	2	
	64 bins or log data input and destination is TRAM or OPRAM	4	
COPY (If preprocessing==mag then x=1, else x=0)	Transpose	x + 11	(Vec_len/8) + x + 11
	8x4 Transpose	x + 8	(vec_len/8) + x + 8
	Copy unpack	x + 4	(2*Vec_len/8) + x + 4
	Real pack or imaginary	x + 4	(Vec_len/8) + x + 4
	Partial copy real or imaginary	x + 4	(Vec_len/8) + x + 4
	Partial copy (real to imag & imag to real)	x + 4	(Vec_len/8) + x + 4
	Copy clear	x + 2	(Vec_len/8) + x + 2
	Shift	x + 5	(Vec_len/8) + x + 5
	Simple copy	x + 4	(Vec_len/8) + x + 4
	Threshold copy	x + 7	(Vec_len/8) + x + 7
MAXS (if preprocessing == MAG, then x=1 else x=0) (if IN_PACK == 2'b10, then p =16 else p =8)	maxs_en = 1 and global max	x + 7	(Vec_len/p) + 7 + x
	maxs_en = 1 with local bitfield and vector length not multiple of 16	x + 7	(Vec_len/p) + 7 + x
	maxs_en = 0 and local max with threshold	x + 9	(Vec_len/p) + 9 + x
	For all other cases	x + 8	(Vec_len/p) + 8 + x
WIN	All cases	9	(Vec_len/8) + 9
RDX2 and IRDX2(real_fft option not available in IRDX2)	Vector length =8 and clubbed fft8 and real_fft for vec_len=8	11	(Vec_len/8) + 11
	All other cases, including real fft	12	(Vec_len/8) + 12
FIR	All cases	9	(Vec_len) + 9
SCP	All cases	9	(Vec_len/8) + 9

Table continues on the next page...

Table 45-72. Table showing pipelining cycles and total number of cycles for operation commands' implementation (continued)

Command	Mode	No. of additional pipeline cycles	Total no. of cycles (throughput)
RDX4 and IRDX4	Round 0 and vector length = 8	10	(Vec_len/8) + 10
	Round 0 and vector length =16	10	(Vec_len/8) + 10
	Round 0 and vector length =32 and clubbed fft32	12	(Vec_len/8) + 12
	Round 0 and vector length =64	12	(Vec_len/8) + 12
	Round 0 and clubbed fft16 and clubbed fft8	10	(Vec_len/8) + 10
	Round 1	12	(Vec_len/8) + 12
	Round > 0 and clubbed fft16	12	(Vec_len/8) + 12
	All other cases	16	(Vec_len/8) + 16
VMT	Operation Sequence 1- Mag2-log2	4+4	(Vec_len/8) +4+4
	Operation Sequence 2 - Stage1 - Abs	3+1	(Vec_len/8) + 3+1
	Operation Sequence 2 - Stage1 - Mag	3+2	(Vec_len/8) + 3+2
	Operation Sequence 2 - Stage1 - Conj	3+1	(Vec_len/8) + 3+1
	Operation Sequence 2 - Stage2 - Shift	3+2+x(x=1 is pre-processing is mag, else x=0)	(Vec_len/8) + 3+2+x
	Operation Sequence 2 - Stage2 - Offset	3+2+x	(Vec_len/8) + 3+2+x
	Operation Sequence 2 - Stage2 - Vector sum	3+2+x	(Vec_len/8) + 3+2+x
	Operation Sequence 2 - Stage3 - Sum and scale	3+4+x	(Vec_len/8) + 3+4+x

The table given below shows the throughput available in different modes of the PDMA command

Table 45-73. Throughput in various modes of PDMA

Mode	Additional cycles	Effective chunk length	Throughput (number of cycles for entire command)
Transfer from SPT RAM to SRAM			
Complex 16-bit	Init Delay = 24+7*mult_clock Final Delay = 7+2*mult_clock + 3*mult_clock_system additional cycles = Init Delay + Final Delay	2	(Vec_len/effective chunk length) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait
Complex 24-bit		1	
Real 16-bit		2	
Real 24-bit		4	

Table continues on the next page...

Table 45-73. Throughput in various modes of PDMA (continued)

Mode	Additional cycles	Effective chunk length	Throughput (number of cycles for entire command)
Binary 48-bit		1	
Complex 16-bit swap		2	
CP4		8	
CP6		8	
CP8		8	
CP16		4	
CP4D		4	
Index val aggregate		1	
Index cross sum aggregate		1	
Mix Mode		2	
CP4FMTB		8	(Vec_len/effective chunk length + no_of_exponent) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait + fmtb_mantissa_exponent_swith_delay*mult_clock
CP4DFMTB		4	
CP8FMTB		4	
CP16FMTB		4	
Absolute aggregate		1	(Vec_len/effective chunk length) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait + (18 * mult_clk)
CP4FMTA	Init Delay = 40+7*mult_clock Final Delay = 7+2*mult_clock + 3*mult_clock_system additional cycles = Init Delay + Final Delay	16	(Vec_len/effective chunk length) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait
CP4DFMTA		8	
Transfer from SRAM to SPT RAM			
Complex 16-bit	Init Delay = 3+7*mult_clock Final Delay = 11+4*mult_clock+ 3*mult_clock_system additional cycles = Init Delay + Final Delay	1	(Vec_len/effective chunk length) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait
Complex 24-bit		1	
Real 16-bit		1	
Real 24-bit		1	
Binary 48-bit		1	
Complex 16-bit swap		1	
CP4		1	
CP6		1	
CP8		1	
CP4D		1	
Mix Mode		1	
CP4FMTA		1	
CP4DFMTA		1	

Table continues on the next page...

Table 45-73. Throughput in various modes of PDMA (continued)

Mode	Additional cycles	Effective chunk length	Throughput (number of cycles for entire command)
CP4FMTB		1	(Vec_len/effective chunk length+no_of_exponent) * mult_clock + additional cycles + hready wait + pdma priority wait + pdma access wait + fmtb_mantissa_exponent_switch_delay*mult_clock
CP4DFMTB		1	
CP8FMTB		1	
CP16FMTB		1	

NOTE

- hready wait is the number of extra cycles spent when hready is low during the entire pdma transfer
- pdma priority wait is the number of extra cycles spent when CS fetches data during a pdma transfer
- pdma access wait is the number of cycles in which access is granted to PDMA
- mult_clock = (ahb clock period/spt clock period)
- mult_clock_system= (ipg_clock_period/spt_clock_period)
- no_of_exponent = ceiling((Vec_len/effective_chunk_length)/ mantissa_per_exponent)

  mantissa_per_exponent = 16 (for CP4FMTB & CP4DFMTB)

  = 32 (for CP8FMTB)

  = 64 (for CP16FMTB)
- fmtb_mantissa_expoent_switch_delay = 4 AHB cycles per exponent (decompression mode) and 2 AHB cycles per exponent (compression mode) for CP4FMTB, CP4DFMTB, CP8FMTB, CP16FMTB

Chapter 46
Cross Triggering Engine (CTE)

46.1 Chip-specific CTE information

46.1.1 Chip-specific Cross Triggering Engine (CTE) information

CTE defines the individual timing of signals which need to be asserted in the system. It controls system's processing flow by triggering modules in the system like ADC sample capture, WGM start to enable DAC sample generation and SPT program execution for autonomous and coherent data processing. Programmable signals can be provided to eTimer for triggering other functional units.

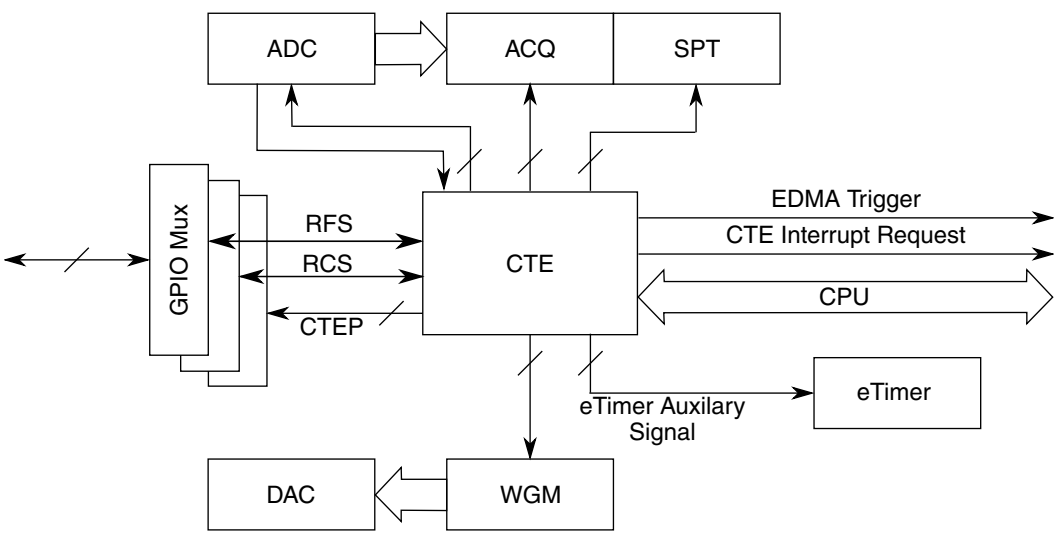


Figure 46-1. CTE Integration

The Radar processing connection between MIPI and CTE as per spec is implemented as shown below:

Input Module	Input Signal	Driving Module	Driver Signal
CTE	RCS[1]	MIPI	HSYNC

Table continues on the next page...

Input Module	Input Signal	Driving Module	Driver Signal
CTE	RFS[2]	MIPI	VSYNC

1. RCS = HSYNC
2. RFS = VSYNC

- Input to the CTE will come from a mux that will choose RCS/RFS either from HSYNC/VSYNC of MIPI or the IO-pads. This is needed because the CTE may use MIPI outputs for generating its timing information.
- Output RCS/RFS of the CTE will be muxed with MIPI's HSYNC/VSYNC and then this mux's outputs will be used by all the other IPs (AFE-filt/WGM/ENET and IO-pads). This is needed to give the timing information synchronously to all the modules.
- MCB_MISC1[IN_CTE] is used to control the input path of RCS/RFS to CTE:
  - 0 - CTE path is selected from PADS.
  - 1 - MIPI path is selected.
- MCB_MISC1[OUT_CTE] is used to control the output path of RCS/RFS from CTE:
  - 0 - CTE path is selected.
  - 1 - MIPI path is selected.

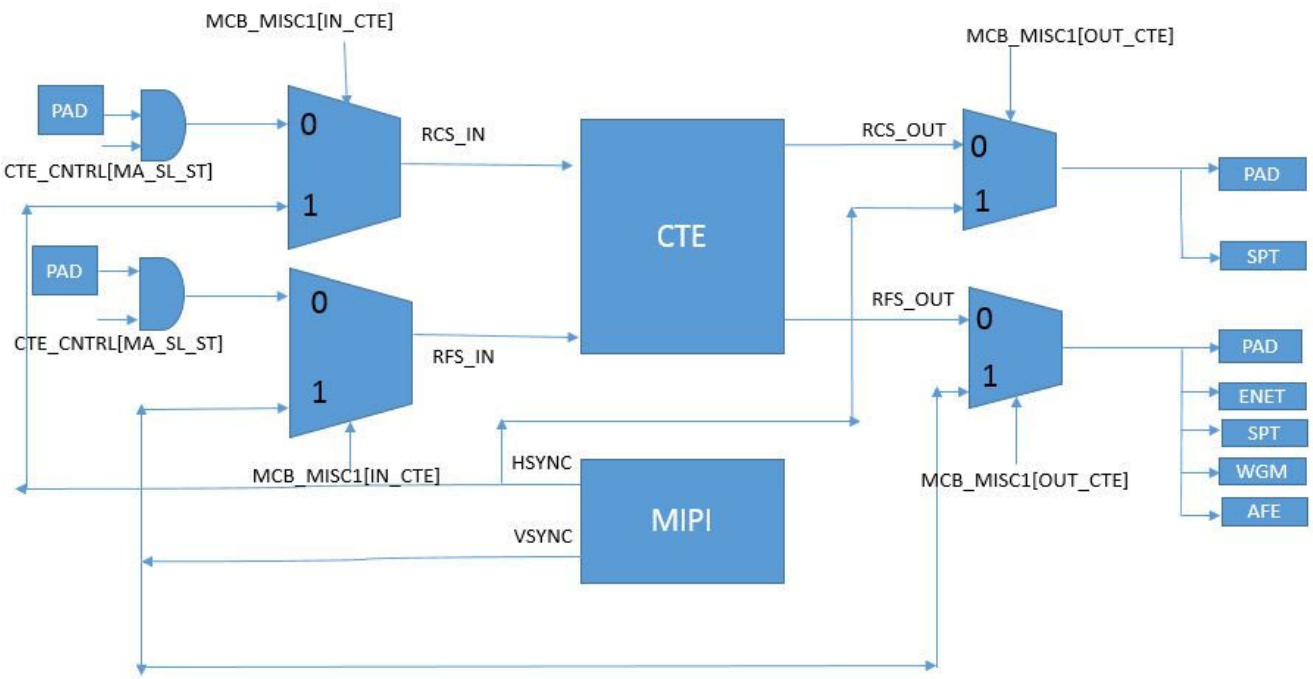


Figure 46-2. Radar processing connection

46.2 Introduction

CTE defines individual timing of signals for RADAR systems and control of the RADAR processing flow. CTE triggers ADC sample capture andDAC conversions as well asSPT program execution for autonomous and coherent data processing. Programmable signals can be provided to various modules in the system, in order to trigger other functional units and for flexible, related timing generation.

The following table lists various acronyms and their description that are used in CTE block.

Acronym	Description
SPT	Signal Processing Toolbox
WGM	Waveform Generator Module
DAC	Digital-to-Analog Converter
LUT	Look-up Table
CTE	Cross Triggering Engine
FSM	Finite State Machine
GPIO	General Purpose I/O
SW	Software
FADR	Fixed Address
CVAL	Counter Value
MISR	Multi Input Shift Register
TT	Timing table. TT0 is first timing table and TT1 is second timing table.

46.3 Block diagram

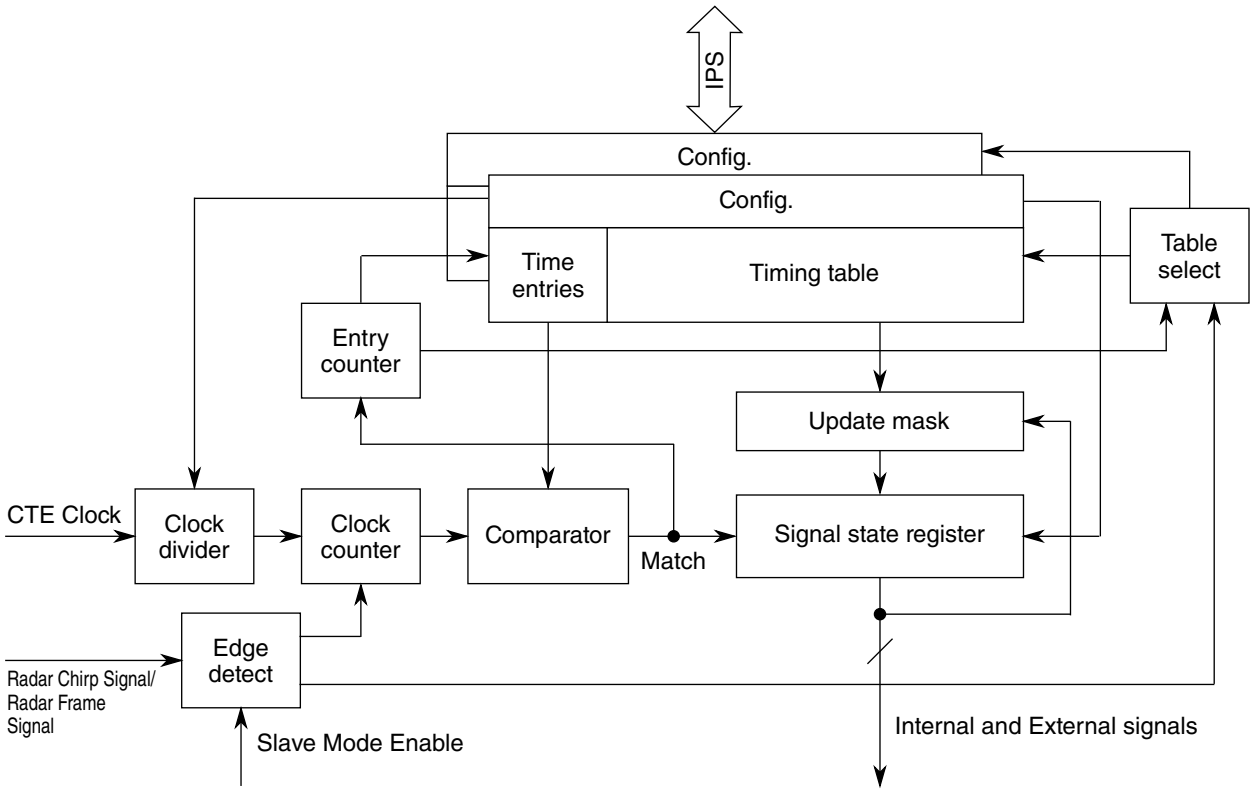


Figure 46-3. CTE block diagram

46.4 Features

CTE module has the following key features:

- Programmable timing generation for RADAR systems
- Synchronization between DAC waveform and ADC capture
- Master/slave mode
- Automatic update by eDMA
- Generation of external timing signals
- Overflow count

46.5 Modes of operation

46.5.1 Master/Slave mode

Master/slave mode operation can be selected using the Control Register master/slave select bit, CNTRL[MA_SL_ST]. By default, CNTRL[MA_SL_ST] = 0 meaning CTE operates in master mode out of reset.

46.5.1.1 Master mode

In master mode operation, the Radar Chirp Synchronization (RCS) and Radar Frame Synchronization (RFS) signals act as outputs for the control of external systems. CTE defines the values of these signals (alongside other external and internal signals) at specific time instances based on internal timing reference look-up tables. The timing tables are populated through the programming of a set of registers, see Timing table configuration section.

46.5.1.2 Slave mode

In slave mode operation, the chirp waveform generation and acquisition unit triggering depends on RCS and RFS as input signals to the CTE. RFS and RCS can be captured from I/O pads or generated by the MIPICSI2 module. Please refer to Chip-specific Cross Triggering Engine (CTE) section, for instructions on how to configure the source of these signals.

The CTE operation mode FSM starts on capture of RCS (after the synchronizer and the delay configuration block). RCS triggers the start of timing table execution, based on the CTE configuration. Capture of RFS (after the synchronizer and the delay configuration block) resets the entry address of timing table execution to ensure that the timing table execution in slave mode is started from location 0. The entry address is automatically reset to 0 after each timing table has finished execution, meaning that external RFS signal is not necessary when it can be ensured that there is enough time between each external RCS to finish timing table execution. Since each captured RCS signal triggers a timing table execution, each timing table should be programmed to represent a single chirp.

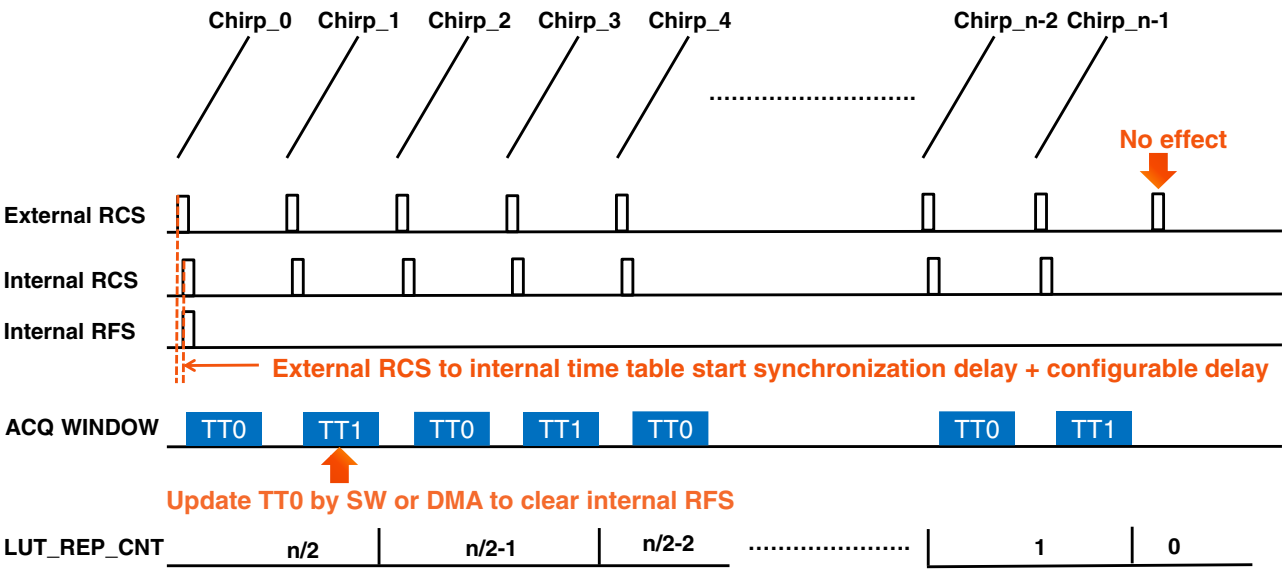


Figure 46-4. Fast chirp modulation timing – example

The internal RFS and RCS still have to be generated as in master mode, meaning the values and behavior of RCS and RFS must be programmed in the timing table registers. They have the same functionality in master and slave mode apart from the functionality that is taken care of by the external signals in slave mode as described above.

From the user perspective, RCS capturing simultaneously or after RFS is the start of the table execution and the next RCS is considered as the end. The end is also dependent upon the duration counter (CTE_LUT_DUR[TT0_DUR]/CTE_LUT_DUR1[TT1_DUR]) and this takes precedence over RCS for considering the end of execution. For example, if the next RCS comes before the TT execution ends, it is ignored. Therefore, it is the responsibility of SW to ensure that the table execution time is always less than the time difference between the two RCS signals. On the contrary, if the table execution is completed before the RCS, signal states will be driven according to the last timing entry until the duration counter expires or the next RCS is captured. Before starting a new frame after all chirps have been processed, the CTE module must be disabled then enabled to reset the FSM.

46.5.1.2.1 Capturing RCS and RFS signals

In order to capture the RCS and RFS pulses correctly, the width of these pulses must be at least twice the clock period of the CTE clock. If the pulse width is less than this, slave mode operation will not work properly. After this 2 flop synchronization, there is an edge detection block which detects the rising edge of the synchronized RCS and RFS signals.

The edge detection block is configurable and can introduce 0-15 clock cycles delay between the input and output signal using CNTRL[RFS_DLY] for RFS and CNTRL[RCS_DLY] for RCS. By adjusting these delays, CPU can control the timing relation between RCS and RFS signals.

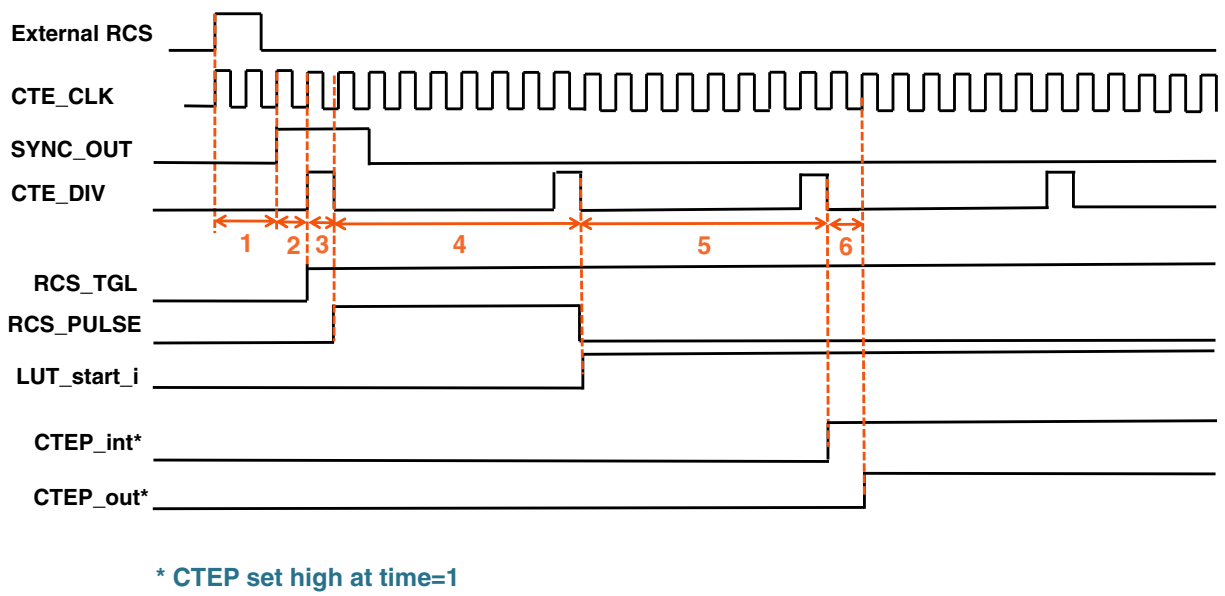


Figure 46-5. External RCS to CTE time-table start Delay_Best_Case

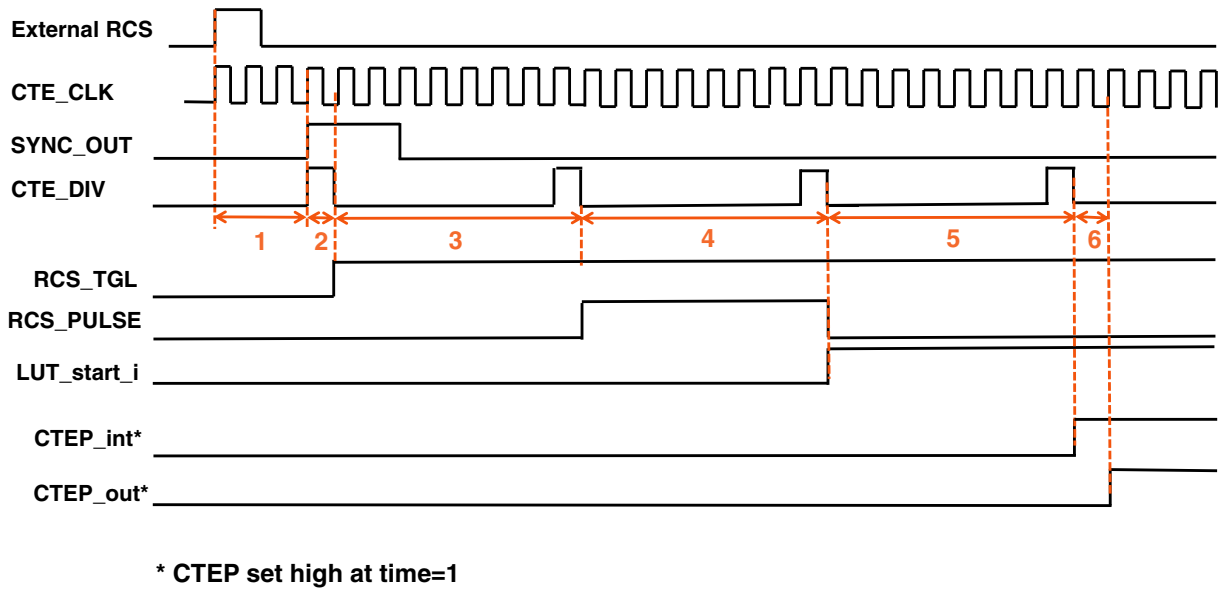


Figure 46-6. External RCS to CTE time-table Start Delay_Worst_Case

1. External RCS passes 2-flop synchronizer running on 80 MHz CTE clock (SDPLL CLK80): 2…3 x 12.5 ns = 25 ns…37.5 ns
2. RCS toggle detection is synchronized to 10/20/40/80 MHz CTE_DIV clock with 12.5% duty cycle, depending on clock selection – any edge(one CLK80 period).
3. RCS pulse is detected with CTE_DIV clock after RCS toggle is detected – negative edge only.
4. One CTE_DIV clock cycle (10/20/40/80 MHz) cycle to start time-table: 100/50/25/12.5 ns.
5. One CTE_DIV clock cycle (10/20/40/80 MHz) cycle to reach time instance 1: 100/50/25/12.5 ns.
6. One CTE clock cycle for output buffering: 12.5 ns.

NOTE

2 and 3 above can overall vary two to nine CTE clock cycles: 25 ns … 112.5 ns.

Additional delay of 0…15 CTE clock cycles (80 MHz) can be introduced via CTE_CNTRL register bit fields RCS_DLY and RFS_DLY.

46.6 Signal description

46.6.1 Signal description

The following diagram depicts all the signals of CTE module.

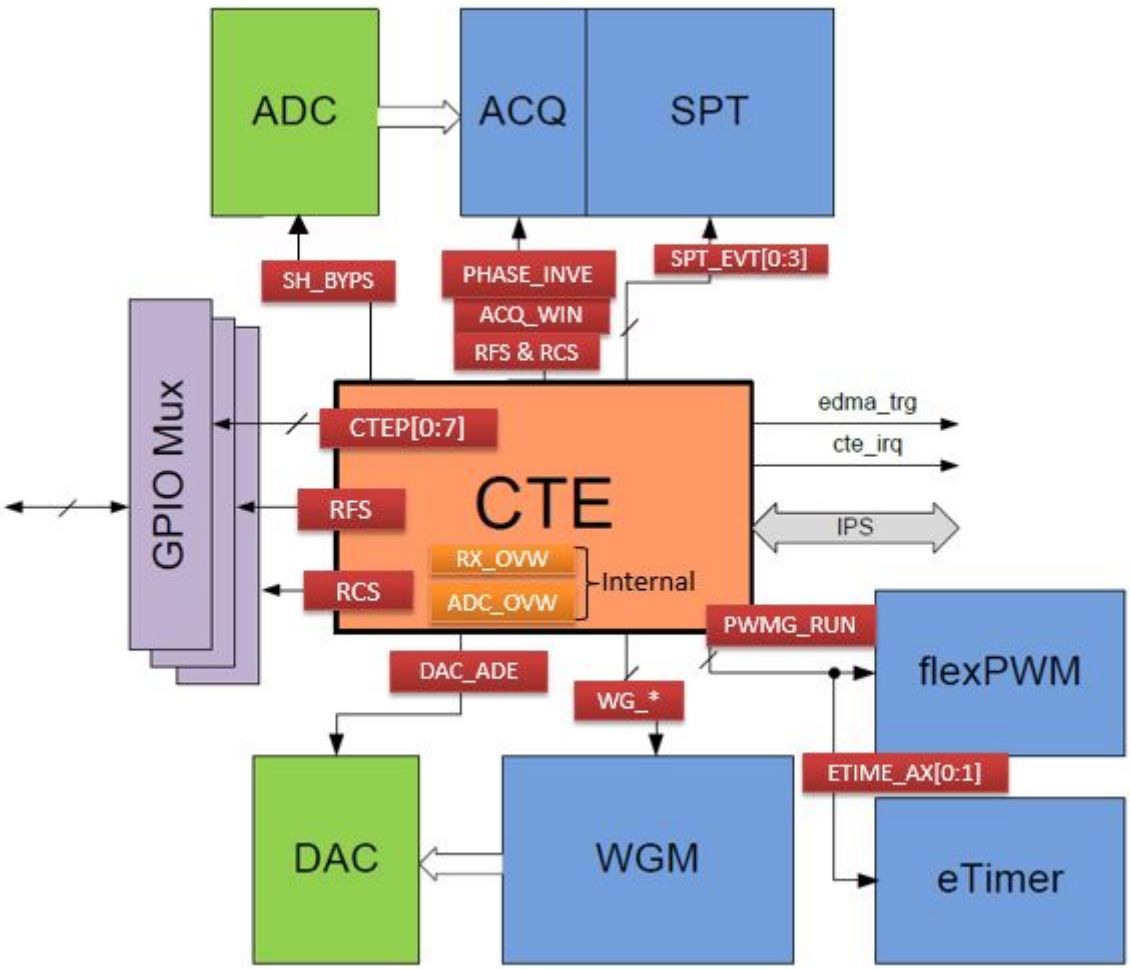


Figure 46-7. CTE signals

46.6.2 Type of external timing signals

The following table shows the list of external timing signals.

Table 46-1. External timing signals

SNo	Signal
1	CTEP[7:0][1]
2	RFS[2]
3	RCS[3]

1. These 8 bits are called CTE pulses. These pulses can be generated by the CTE block based on the timing table registers.
2. Radar frame synchronization is output in case of master mode and input in case of slave mode. This signal indicates the start of a frame. In slave mode, this signal needs to be synchronized and sampled to detect the start of a frame.
3. Radar chirp synchronization is output in case of master mode and input in case of slave mode. This signal indicates the start of a chirp within frame. In slave mode, this signal needs to be synchronized and sampled to detect the start of a chirp.

The type of external timing signal can be selected by configuring the bits corresponding to each signals in signal type0 Signal Type Register 0 (CTE_SIGTYPE0*n*) and signal type1 Signal Type Register 1 (CTE_SIGTYPE1*n*) registers.

External timing signals can be one of the following types.

46.6.2.1 Logic

Logic type is the simplest definition. Signals are asserted/de-asserted at defined timing instances in the timing table register LUT_MSB or LUT_LSB, depending on the values shown below. Signals having value as 11b are considered as don't care and do not change signal state.

- 00b: set signal to low level
- 01b: set signal to high level
- 10b: set signal to high-Z
- 11b: don't care (no change)

The signal does not reset when the last entry in the timing table is reached, but keeps its level until the next definition is reached.

46.6.2.2 Toggle

Toggle type is similar to logic type, but defines the signal state change instead of the actual level. With toggle type, the signal changes its state to the inverse with respect to the previous state when signal definition 10b was defined in LUT_MSB register for that particular signal. Value 2'b11 has no effect (don't care) and is used to skip time entries which are defined for logic signals, but shall not influence the toggle signal.

- 00b: reset signal to low level
- 01b: reset signal to high level
- 10b: toggle signal (flip level)
- 11b: don't care (no change)

A toggle signal keeps its state after the last entry in the timing table was reached until it is modified by an entry in following timing tables.

46.6.2.3 Clock

A clock signal toggles periodically with a defined period in control register1 at address 0x0004. Timing table entries define the active and inactive periods of this generated clock signal. Clock generation is started when the first entry in the timing table is 10b. If a timing entry other than 01b does not hit a corresponding subsequent generated clock edge, i.e. occurs during low or high time, the clock period is shortened or prolonged, depending on the previous state of the clock generator because entry 01b causes the rising edge of the clock if clock value is 0. If clock value is 1, entry 01b will keep it as 1.

- 00b: clock held to low
- 01b: clock active (toggling and synchronize rising edge)
- 10b: clock active (toggling)
- 11b: clock held high

See Figure 46-11 for more details.

Four independent clock counters are available for use with these clock signals. The toggle rate is defined by the divider ratio for each clock counter. The clock divider ratio is an integer number and can be defined once for each timing table as described in the control register1 CTE Control Register 1 (CTE_CNTRL1).

46.6.3 Signal configuration example

The following figure shows an example of how the signal definition can be configured for various external signals.

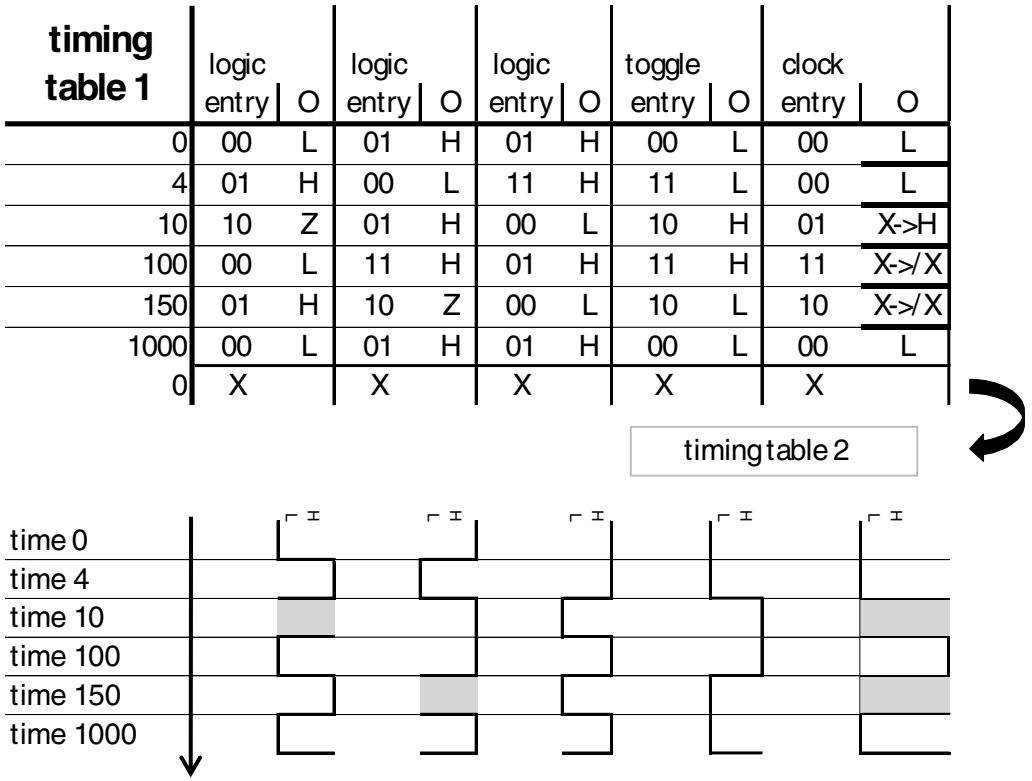


Figure 46-8. External signal configuration example

The following figure shows an example of how the signal definition can be configured for various internal signals.

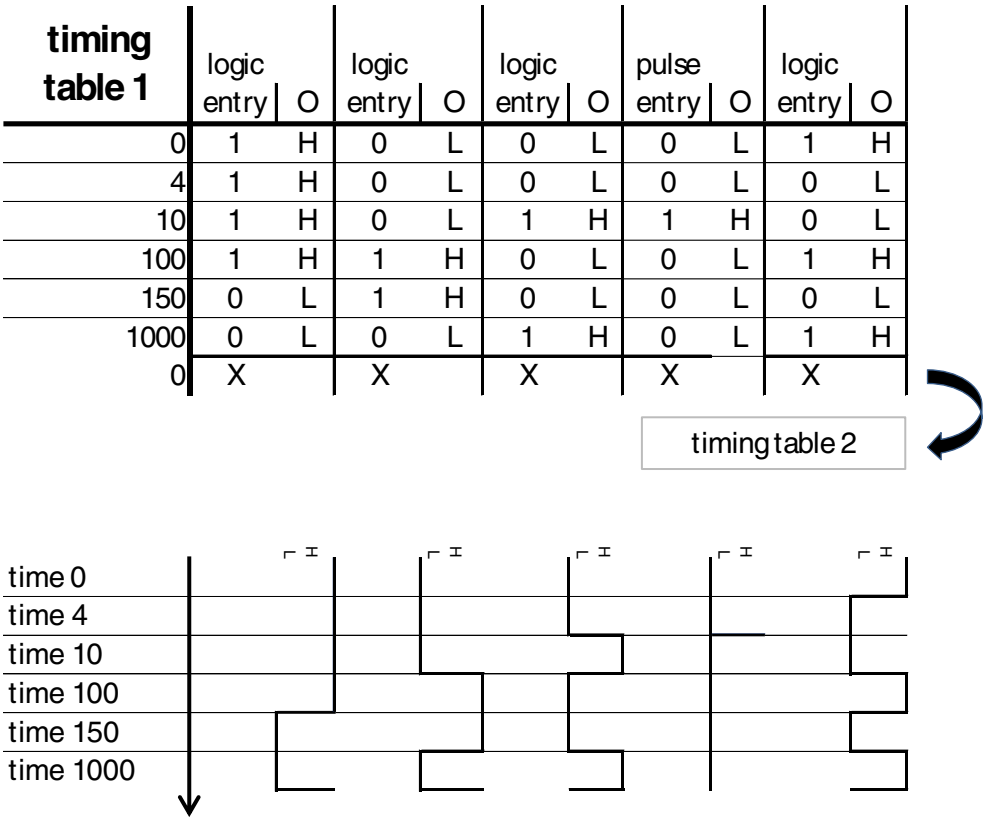


Figure 46-9. Internal signal configuration example

46.7 Memory map and register definition

The CTE is programmed by an asynchronous CPU interface. Additionally, timing tables can also be configured using the eDMA. The registers are 32 bit wide, however it shall be possible to program single bytes and words in the timing table registers without changing the other bytes of same register (masked write). It contains the interrupt mask and status, eDMA control registers and the configuration registers for the timing tables, including the timing values and assigned signal states. The register with timing values is valid for both internal and external signals. The timing values shall be 16-bit aligned.

CTE memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (CTE_CNTRL)	32	R/W	0000_0000h	46.7.1/2114
4	CTE Control Register 1 (CTE_CNTRL1)	32	R/W	0000_0000h	46.7.2/2116
8	First Timing Table Register (LSB) (CTE_LUT0_LSB0)	32	R/W	0000_0000h	46.7.3/2119
C	First Timing Table Register (LSB) (CTE_LUT0_LSB1)	32	R/W	0000_0000h	46.7.3/2119
10	First Timing Table Register (LSB) (CTE_LUT0_LSB2)	32	R/W	0000_0000h	46.7.3/2119

Table continues on the next page...

CTE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14	First Timing Table Register (LSB) (CTE_LUT0_LSB3)	32	R/W	0000_0000h	46.7.3/2119
18	First Timing Table Register (LSB) (CTE_LUT0_LSB4)	32	R/W	0000_0000h	46.7.3/2119
1C	First Timing Table Register (LSB) (CTE_LUT0_LSB5)	32	R/W	0000_0000h	46.7.3/2119
20	First Timing Table Register (LSB) (CTE_LUT0_LSB6)	32	R/W	0000_0000h	46.7.3/2119
24	First Timing Table Register (LSB) (CTE_LUT0_LSB7)	32	R/W	0000_0000h	46.7.3/2119
28	First Timing Table Register (LSB) (CTE_LUT0_LSB8)	32	R/W	0000_0000h	46.7.3/2119
2C	First Timing Table Register (LSB) (CTE_LUT0_LSB9)	32	R/W	0000_0000h	46.7.3/2119
30	First Timing Table Register (LSB) (CTE_LUT0_LSB10)	32	R/W	0000_0000h	46.7.3/2119
34	First Timing Table Register (LSB) (CTE_LUT0_LSB11)	32	R/W	0000_0000h	46.7.3/2119
38	First Timing Table Register (LSB) (CTE_LUT0_LSB12)	32	R/W	0000_0000h	46.7.3/2119
3C	First Timing Table Register (LSB) (CTE_LUT0_LSB13)	32	R/W	0000_0000h	46.7.3/2119
40	First Timing Table Register (LSB) (CTE_LUT0_LSB14)	32	R/W	0000_0000h	46.7.3/2119
44	First Timing Table Register (LSB) (CTE_LUT0_LSB15)	32	R/W	0000_0000h	46.7.3/2119
48	First Timing Table Register (LSB) (CTE_LUT0_LSB16)	32	R/W	0000_0000h	46.7.3/2119
4C	First Timing Table Register (LSB) (CTE_LUT0_LSB17)	32	R/W	0000_0000h	46.7.3/2119
50	First Timing Table Register (LSB) (CTE_LUT0_LSB18)	32	R/W	0000_0000h	46.7.3/2119
54	First Timing Table Register (LSB) (CTE_LUT0_LSB19)	32	R/W	0000_0000h	46.7.3/2119
58	First Timing Table Register (LSB) (CTE_LUT0_LSB20)	32	R/W	0000_0000h	46.7.3/2119
5C	First Timing Table Register (LSB) (CTE_LUT0_LSB21)	32	R/W	0000_0000h	46.7.3/2119
60	First Timing Table Register (LSB) (CTE_LUT0_LSB22)	32	R/W	0000_0000h	46.7.3/2119
64	First Timing Table Register (LSB) (CTE_LUT0_LSB23)	32	R/W	0000_0000h	46.7.3/2119
68	First Timing Table Register (LSB) (CTE_LUT0_LSB24)	32	R/W	0000_0000h	46.7.3/2119
6C	First Timing Table Register (LSB) (CTE_LUT0_LSB25)	32	R/W	0000_0000h	46.7.3/2119
70	First Timing Table Register (LSB) (CTE_LUT0_LSB26)	32	R/W	0000_0000h	46.7.3/2119
74	First Timing Table Register (LSB) (CTE_LUT0_LSB27)	32	R/W	0000_0000h	46.7.3/2119
78	First Timing Table Register (LSB) (CTE_LUT0_LSB28)	32	R/W	0000_0000h	46.7.3/2119
7C	First Timing Table Register (LSB) (CTE_LUT0_LSB29)	32	R/W	0000_0000h	46.7.3/2119
80	First Timing Table Register (LSB) (CTE_LUT0_LSB30)	32	R/W	0000_0000h	46.7.3/2119
84	First Timing Table Register (LSB) (CTE_LUT0_LSB31)	32	R/W	0000_0000h	46.7.3/2119
88	First Timing Table Register (MSB) (CTE_LUT0_MSB0)	32	R/W	0000_0000h	46.7.4/2121
8C	First Timing Table Register (MSB) (CTE_LUT0_MSB1)	32	R/W	0000_0000h	46.7.4/2121
90	First Timing Table Register (MSB) (CTE_LUT0_MSB2)	32	R/W	0000_0000h	46.7.4/2121
94	First Timing Table Register (MSB) (CTE_LUT0_MSB3)	32	R/W	0000_0000h	46.7.4/2121
98	First Timing Table Register (MSB) (CTE_LUT0_MSB4)	32	R/W	0000_0000h	46.7.4/2121
9C	First Timing Table Register (MSB) (CTE_LUT0_MSB5)	32	R/W	0000_0000h	46.7.4/2121
A0	First Timing Table Register (MSB) (CTE_LUT0_MSB6)	32	R/W	0000_0000h	46.7.4/2121
A4	First Timing Table Register (MSB) (CTE_LUT0_MSB7)	32	R/W	0000_0000h	46.7.4/2121
A8	First Timing Table Register (MSB) (CTE_LUT0_MSB8)	32	R/W	0000_0000h	46.7.4/2121
AC	First Timing Table Register (MSB) (CTE_LUT0_MSB9)	32	R/W	0000_0000h	46.7.4/2121

Table continues on the next page...

CTE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B0	First Timing Table Register (MSB) (CTE_LUT0_MSB10)	32	R/W	0000_0000h	46.7.4/2121
B4	First Timing Table Register (MSB) (CTE_LUT0_MSB11)	32	R/W	0000_0000h	46.7.4/2121
B8	First Timing Table Register (MSB) (CTE_LUT0_MSB12)	32	R/W	0000_0000h	46.7.4/2121
BC	First Timing Table Register (MSB) (CTE_LUT0_MSB13)	32	R/W	0000_0000h	46.7.4/2121
C0	First Timing Table Register (MSB) (CTE_LUT0_MSB14)	32	R/W	0000_0000h	46.7.4/2121
C4	First Timing Table Register (MSB) (CTE_LUT0_MSB15)	32	R/W	0000_0000h	46.7.4/2121
C8	First Timing Table Register (MSB) (CTE_LUT0_MSB16)	32	R/W	0000_0000h	46.7.4/2121
CC	First Timing Table Register (MSB) (CTE_LUT0_MSB17)	32	R/W	0000_0000h	46.7.4/2121
D0	First Timing Table Register (MSB) (CTE_LUT0_MSB18)	32	R/W	0000_0000h	46.7.4/2121
D4	First Timing Table Register (MSB) (CTE_LUT0_MSB19)	32	R/W	0000_0000h	46.7.4/2121
D8	First Timing Table Register (MSB) (CTE_LUT0_MSB20)	32	R/W	0000_0000h	46.7.4/2121
DC	First Timing Table Register (MSB) (CTE_LUT0_MSB21)	32	R/W	0000_0000h	46.7.4/2121
E0	First Timing Table Register (MSB) (CTE_LUT0_MSB22)	32	R/W	0000_0000h	46.7.4/2121
E4	First Timing Table Register (MSB) (CTE_LUT0_MSB23)	32	R/W	0000_0000h	46.7.4/2121
E8	First Timing Table Register (MSB) (CTE_LUT0_MSB24)	32	R/W	0000_0000h	46.7.4/2121
EC	First Timing Table Register (MSB) (CTE_LUT0_MSB25)	32	R/W	0000_0000h	46.7.4/2121
F0	First Timing Table Register (MSB) (CTE_LUT0_MSB26)	32	R/W	0000_0000h	46.7.4/2121
F4	First Timing Table Register (MSB) (CTE_LUT0_MSB27)	32	R/W	0000_0000h	46.7.4/2121
F8	First Timing Table Register (MSB) (CTE_LUT0_MSB28)	32	R/W	0000_0000h	46.7.4/2121
FC	First Timing Table Register (MSB) (CTE_LUT0_MSB29)	32	R/W	0000_0000h	46.7.4/2121
100	First Timing Table Register (MSB) (CTE_LUT0_MSB30)	32	R/W	0000_0000h	46.7.4/2121
104	First Timing Table Register (MSB) (CTE_LUT0_MSB31)	32	R/W	0000_0000h	46.7.4/2121
108	Second Timing Table Register (LSB) (CTE_LUT1_LSB0)	32	R/W	0000_0000h	46.7.5/2123
10C	Second Timing Table Register (LSB) (CTE_LUT1_LSB1)	32	R/W	0000_0000h	46.7.5/2123
110	Second Timing Table Register (LSB) (CTE_LUT1_LSB2)	32	R/W	0000_0000h	46.7.5/2123
114	Second Timing Table Register (LSB) (CTE_LUT1_LSB3)	32	R/W	0000_0000h	46.7.5/2123
118	Second Timing Table Register (LSB) (CTE_LUT1_LSB4)	32	R/W	0000_0000h	46.7.5/2123
11C	Second Timing Table Register (LSB) (CTE_LUT1_LSB5)	32	R/W	0000_0000h	46.7.5/2123
120	Second Timing Table Register (LSB) (CTE_LUT1_LSB6)	32	R/W	0000_0000h	46.7.5/2123
124	Second Timing Table Register (LSB) (CTE_LUT1_LSB7)	32	R/W	0000_0000h	46.7.5/2123
128	Second Timing Table Register (LSB) (CTE_LUT1_LSB8)	32	R/W	0000_0000h	46.7.5/2123
12C	Second Timing Table Register (LSB) (CTE_LUT1_LSB9)	32	R/W	0000_0000h	46.7.5/2123
130	Second Timing Table Register (LSB) (CTE_LUT1_LSB10)	32	R/W	0000_0000h	46.7.5/2123
134	Second Timing Table Register (LSB) (CTE_LUT1_LSB11)	32	R/W	0000_0000h	46.7.5/2123
138	Second Timing Table Register (LSB) (CTE_LUT1_LSB12)	32	R/W	0000_0000h	46.7.5/2123
13C	Second Timing Table Register (LSB) (CTE_LUT1_LSB13)	32	R/W	0000_0000h	46.7.5/2123
140	Second Timing Table Register (LSB) (CTE_LUT1_LSB14)	32	R/W	0000_0000h	46.7.5/2123
144	Second Timing Table Register (LSB) (CTE_LUT1_LSB15)	32	R/W	0000_0000h	46.7.5/2123
148	Second Timing Table Register (LSB) (CTE_LUT1_LSB16)	32	R/W	0000_0000h	46.7.5/2123

Table continues on the next page...

CTE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14C	Second Timing Table Register (LSB) (CTE_LUT1_LSB17)	32	R/W	0000_0000h	46.7.5/2123
150	Second Timing Table Register (LSB) (CTE_LUT1_LSB18)	32	R/W	0000_0000h	46.7.5/2123
154	Second Timing Table Register (LSB) (CTE_LUT1_LSB19)	32	R/W	0000_0000h	46.7.5/2123
158	Second Timing Table Register (LSB) (CTE_LUT1_LSB20)	32	R/W	0000_0000h	46.7.5/2123
15C	Second Timing Table Register (LSB) (CTE_LUT1_LSB21)	32	R/W	0000_0000h	46.7.5/2123
160	Second Timing Table Register (LSB) (CTE_LUT1_LSB22)	32	R/W	0000_0000h	46.7.5/2123
164	Second Timing Table Register (LSB) (CTE_LUT1_LSB23)	32	R/W	0000_0000h	46.7.5/2123
168	Second Timing Table Register (LSB) (CTE_LUT1_LSB24)	32	R/W	0000_0000h	46.7.5/2123
16C	Second Timing Table Register (LSB) (CTE_LUT1_LSB25)	32	R/W	0000_0000h	46.7.5/2123
170	Second Timing Table Register (LSB) (CTE_LUT1_LSB26)	32	R/W	0000_0000h	46.7.5/2123
174	Second Timing Table Register (LSB) (CTE_LUT1_LSB27)	32	R/W	0000_0000h	46.7.5/2123
178	Second Timing Table Register (LSB) (CTE_LUT1_LSB28)	32	R/W	0000_0000h	46.7.5/2123
17C	Second Timing Table Register (LSB) (CTE_LUT1_LSB29)	32	R/W	0000_0000h	46.7.5/2123
180	Second Timing Table Register (LSB) (CTE_LUT1_LSB30)	32	R/W	0000_0000h	46.7.5/2123
184	Second Timing Table Register (LSB) (CTE_LUT1_LSB31)	32	R/W	0000_0000h	46.7.5/2123
188	Second Timing Table Register (MSB) (CTE_LUT1_MSB0)	32	R/W	0000_0000h	46.7.6/2125
18C	Second Timing Table Register (MSB) (CTE_LUT1_MSB1)	32	R/W	0000_0000h	46.7.6/2125
190	Second Timing Table Register (MSB) (CTE_LUT1_MSB2)	32	R/W	0000_0000h	46.7.6/2125
194	Second Timing Table Register (MSB) (CTE_LUT1_MSB3)	32	R/W	0000_0000h	46.7.6/2125
198	Second Timing Table Register (MSB) (CTE_LUT1_MSB4)	32	R/W	0000_0000h	46.7.6/2125
19C	Second Timing Table Register (MSB) (CTE_LUT1_MSB5)	32	R/W	0000_0000h	46.7.6/2125
1A0	Second Timing Table Register (MSB) (CTE_LUT1_MSB6)	32	R/W	0000_0000h	46.7.6/2125
1A4	Second Timing Table Register (MSB) (CTE_LUT1_MSB7)	32	R/W	0000_0000h	46.7.6/2125
1A8	Second Timing Table Register (MSB) (CTE_LUT1_MSB8)	32	R/W	0000_0000h	46.7.6/2125
1AC	Second Timing Table Register (MSB) (CTE_LUT1_MSB9)	32	R/W	0000_0000h	46.7.6/2125
1B0	Second Timing Table Register (MSB) (CTE_LUT1_MSB10)	32	R/W	0000_0000h	46.7.6/2125
1B4	Second Timing Table Register (MSB) (CTE_LUT1_MSB11)	32	R/W	0000_0000h	46.7.6/2125
1B8	Second Timing Table Register (MSB) (CTE_LUT1_MSB12)	32	R/W	0000_0000h	46.7.6/2125
1BC	Second Timing Table Register (MSB) (CTE_LUT1_MSB13)	32	R/W	0000_0000h	46.7.6/2125
1C0	Second Timing Table Register (MSB) (CTE_LUT1_MSB14)	32	R/W	0000_0000h	46.7.6/2125
1C4	Second Timing Table Register (MSB) (CTE_LUT1_MSB15)	32	R/W	0000_0000h	46.7.6/2125
1C8	Second Timing Table Register (MSB) (CTE_LUT1_MSB16)	32	R/W	0000_0000h	46.7.6/2125
1CC	Second Timing Table Register (MSB) (CTE_LUT1_MSB17)	32	R/W	0000_0000h	46.7.6/2125
1D0	Second Timing Table Register (MSB) (CTE_LUT1_MSB18)	32	R/W	0000_0000h	46.7.6/2125
1D4	Second Timing Table Register (MSB) (CTE_LUT1_MSB19)	32	R/W	0000_0000h	46.7.6/2125
1D8	Second Timing Table Register (MSB) (CTE_LUT1_MSB20)	32	R/W	0000_0000h	46.7.6/2125
1DC	Second Timing Table Register (MSB) (CTE_LUT1_MSB21)	32	R/W	0000_0000h	46.7.6/2125
1E0	Second Timing Table Register (MSB) (CTE_LUT1_MSB22)	32	R/W	0000_0000h	46.7.6/2125
1E4	Second Timing Table Register (MSB) (CTE_LUT1_MSB23)	32	R/W	0000_0000h	46.7.6/2125

Table continues on the next page...

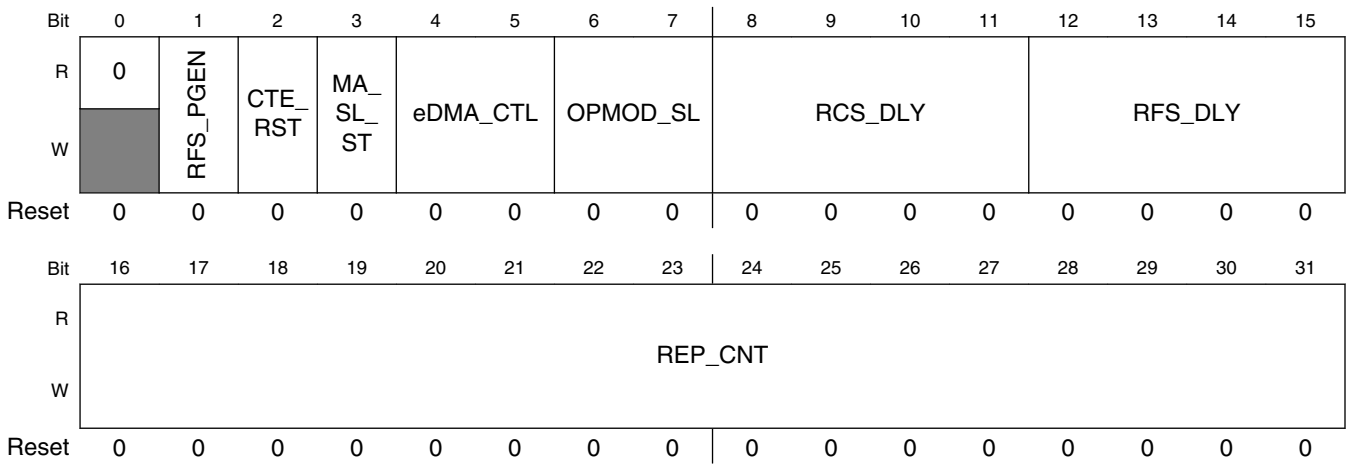
CTE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E8	Second Timing Table Register (MSB) (CTE_LUT1_MSB24)	32	R/W	0000_0000h	46.7.6/2125
1EC	Second Timing Table Register (MSB) (CTE_LUT1_MSB25)	32	R/W	0000_0000h	46.7.6/2125
1F0	Second Timing Table Register (MSB) (CTE_LUT1_MSB26)	32	R/W	0000_0000h	46.7.6/2125
1F4	Second Timing Table Register (MSB) (CTE_LUT1_MSB27)	32	R/W	0000_0000h	46.7.6/2125
1F8	Second Timing Table Register (MSB) (CTE_LUT1_MSB28)	32	R/W	0000_0000h	46.7.6/2125
1FC	Second Timing Table Register (MSB) (CTE_LUT1_MSB29)	32	R/W	0000_0000h	46.7.6/2125
200	Second Timing Table Register (MSB) (CTE_LUT1_MSB30)	32	R/W	0000_0000h	46.7.6/2125
204	Second Timing Table Register (MSB) (CTE_LUT1_MSB31)	32	R/W	0000_0000h	46.7.6/2125
208	Signal Type Register 0 (CTE_SIGTYPE00)	32	R/W	0000_0000h	46.7.7/2127
20C	Signal Type Register 0 (CTE_SIGTYPE01)	32	R/W	0000_0000h	46.7.7/2127
210	Signal Type Register 1 (CTE_SIGTYPE10)	32	R/W	See section	46.7.8/2131
214	Signal Type Register 1 (CTE_SIGTYPE11)	32	R/W	See section	46.7.8/2131
220	CTE Interrupt Enable Register (CTE_INTEN)	32	R/W	0000_0000h	46.7.9/2132
224	CTE Interrupt Status Register (CTE_INTSTAT)	32	w1c	0000_0000h	46.7.10/ 2134
22C	Receiver Overflow Counter (CTE_RCVOFCNT)	32	R	0000_0000h	46.7.11/ 2136
270	LUT Checksum Register (CTE_CKSM_LSB)	32	R/W	0000_0000h	46.7.12/ 2136
274	LUT Checksum Register (CTE_CKSM_MSB)	32	R/W	0000_0000h	46.7.13/ 2137
278	Debug Register (CTE_DBG_REG)	32	R/W	0000_0000h	46.7.14/ 2138
27C	TT0 Execution Duration Register (CTE_LUT_DUR)	32	R/W	0000_0000h	46.7.15/ 2139
280	TT1 Execution Duration Register (CTE_LUT_DUR1)	32	R/W	0000_0000h	46.7.16/ 2140
284	Clock Select Register (CTE_CLKSEL)	32	R/W	0000_0000h	46.7.17/ 2142

46.7.1 Control Register (CTE_CNTRL)

This is the first control register of the CTE block for controlling the data path operations and signal behaviors.

Address: 0h base + 0h offset = 0h



CTE_CNTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 RFS_PGEN	Internal SW RFS Pulse Trigger. This bit is to be used in Slave mode only. This bit is used to generate a single CTE clock period wide internal RFS pulse when it is changed from 0 to 1 (Please note that this bit is not self clearing. For generating the second pulse, this bit needs to be reset and then set again by the software). This pulse is used to simulate RFS internally in order to reset the timing table execution and restart from the first entry, as if an external RFS pulse had been received.
2 CTE_RST	CTE synchronous Reset This is a synchronous reset which resets the entire CTE module on the next clock. When this bit is asserted, different CTE modules are synchronously reset on their next respective clocks. 0 CTE module reset is de-asserted. 1 CTE module reset is asserted.
3 MA_SL_ST	master/slave select This bit used to define if the CTE is working in master or slave mode. 0 CTE is in master mode (default mode). 1 CTE is in slave mode.
4–5 eDMA_CTL	eDMA trigger control These bits are used to trigger the eDMA controller for configuring the timing table registers (LUT_LSB and LUT_MSB). The trigger can be generated on reaching start of table, end of the timing table.

Table continues on the next page...

CTE_CNTRL field descriptions (continued)

Field	Description
	00 eDMA trigger generation disabled. 01 eDMA trigger will be generated at the start of timing table. 10 eDMA trigger will be generated at the end of timing table. 11 Not used. eDMA trigger generation is disabled.
6–7 OPMOD_SL	CTE FSM operation mode select. These bits are used to select the mode in which the CTE FSM operates. See the Timing Table Execution section under Functional description for details. **NOTE:** In continuous run mode or toggle mode, if the "REP_CNT" value is 0, the execution happens forever. 00 Halt Mode. In this mode timing generation is stopped. 10 Continuous run mode, Single table. In this mode the timing table is executed for the number of times programmed in the "lut_rep_cnt". 11 Continuous toggle mode, 2 tables. In this mode, the timing tables are toggled for the number of times programmed in the "lut_rep_cnt". 01 Continuous run mode, TT1. In this mode the second timing table (TT1) is executed for the number of times programmed in the "lut_rep_cnt".
8–11 RCS_DLY	Radar Chirp Synchronization Delay Control These bits are used to introduce configurable clock cycle delay in the RCS signal input when CTE is operating in slave mode. Maximum limit is 15 CTE clocks cycles. **NOTE:** In master mode, read write to these bits is supported but these bits will not have any effect on the fuctionality. 0000 No Delay. 0001 1 clock cycle delay needs to be introduced. 0010 2 clock cycle delay needs to be introduced. 0011 3 clock cycle delay needs to be introduced. 0100 4 clock cycle delay needs to be introduced. 0101 5 clock cycle delay needs to be introduced. 0110 6 clock cycle delay needs to be introduced. 0111 7 clock cycle delay needs to be introduced. 1000 8 clock cycle delay needs to be introduced. 1001 9 clock cycle delay needs to be introduced. 1010 10 clock cycle delay needs to be introduced. 1011 11 clock cycle delay needs to be introduced. 1100 12 clock cycle delay needs to be introduced. 1101 13 clock cycle delay needs to be introduced. 1110 14 clock cycle delay needs to be introduced. 1111 15 clock cycle delay needs to be introduced.
12–15 RFS_DLY	Radar Frame Synchronization Delay Control These bits define how much CTE clock delay has to be introduced in the RFS signal in slave mode. Maximum limit is 15 CTE clocks. **NOTE:** In master mode, read write to these bits is supported but these bits will not have any effect on the fuctionality. 0000 No Delay

Table continues on the next page...

CTE_CNTRL field descriptions (continued)

Field	Description
	0001 1 clock cycle delay needs to be introduced. 0010 2 clock cycle delay needs to be introduced. 0011 3 clock cycle delay needs to be introduced. 0100 4 clock cycle delay needs to be introduced. 0101 5 clock cycle delay needs to be introduced. 0110 6 clock cycle delay needs to be introduced. 0111 7 clock cycle delay needs to be introduced. 1000 8 clock cycle delay needs to be introduced. 1001 9 clock cycle delay needs to be introduced. 1010 10 clock cycle delay needs to be introduced. 1011 11 clock cycle delay needs to be introduced. 1100 12 clock cycle delay needs to be introduced. 1101 13 clock cycle delay needs to be introduced. 1110 14 clock cycle delay needs to be introduced. 1111 15 clock cycle delay needs to be introduced.
16–31 REP_CNT	Timing Table Repetition Counter This counter dictates how many times a selected timing table will be executed. If it is 0, the timing table will be executed forever.

46.7.2 CTE Control Register 1 (CTE_CNTRL1)

This is the second control register of the CTE block for controlling the data path operations and signal behaviors.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		0	0	CKSM_RST	CHKSM_MD	CTE_EN	TIMEMODE	0		CLKDIV_4			CLKDIV_3		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		CLKDIV_2			CLKDIV_1			0		CTECK_DV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_CNTRL1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CKSM_RST	Checksum Reset This bit when asserted will synchronously reset the checksum calculation block on the next IPG clock. This block is also reset on RFS. 0 De-asserted. Checksum calculation functions normally. 1 Checksum calculator block will be held in reset from next cycle onwards.
5 CHKSM_MD	Checksum Mode This bit defines the mode in which checksum is calculated. 0 Checksum mode means that the carry on the MSB is dropped (default). 1 MISR mode means that the carry on the MSB is fed back to the LSB of the checksum.
6 CTE_EN	CTE Enable This bit is used to enable the CTE module. **NOTE:** If it is required to re-configure CTE, this bit should be disabled and enabled at the end after all the other configuration has been done. 0 CTE is disabled. When this bit is 0, all the pending interrupts, eDMA request will be de-asserted immediately. Operational mode FSM will come to HALT and all the timing signals will be de-asserted. Internal time counter will be reset and the timing table entry counter will point to the 1st location of TT0 and all the interrupt status bits will be cleared. 1 Timing table execution is enabled. Note that when CTE enable bit is set from 0 to 1 in the middle of operation, the timing table execution will start from address 0.
7 TIMEMODE	Time mode type This bit defines if the absolute value of the time entries in the table need to be taken or if they are relative to the previous entry. The difference in these two modes is shown in Figure 46-13. 0 Relative, internal time counter will reset once it reaches the current timing entry so that the next timing instance becomes relative to the current time instance. 1 Absolute, internal time counter will not be reset hence the signal states are driven at absolute timing instances.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–12 CLKDIV_4	4th Clock divider This value sets the frequency of the 4th clock dividers which will be used to generate the clock type signal. **NOTE:** This value is taken in power of 2 and the divided clock is of 50% duty cycle. 000 Forth clock divider's frequency is equal to CTE clock frequency. 001 Forth clock divider's frequency is equal to half the CTE clock frequency. 111 Forth clock divider's frequency is equal to 1/128 of CTE clock frequency.

Table continues on the next page...

CTE_CNTRL1 field descriptions (continued)

Field	Description
13–15 CLKDIV_3	3rd Clock divider This value sets the frequency of the 3rd clock dividers which will be used to generate the clock type signal. **NOTE:** This value is taken in power of 2 and the divided clock is of 50% duty cycle. 000 Third clock divider's frequency is equal to CTE clock frequency. 001 Third clock divider's frequency is equal to half the CTE clock frequency. 111 Third clock divider's frequency is equal to 1/128 of CTE clock frequency.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–20 CLKDIV_2	2nd Clock divider This value sets the frequency of the 2nd clock dividers which will be used to generate the clock type signal. **NOTE:** This value is taken in power of 2 and the divided clock is of 50% duty cycle. 000 Second clock divider's frequency is equal to CTE clock frequency. 001 Second clock divider's frequency is equal to half of the CTE clock frequency. 111 Second clock divider's frequency is equal to 1/128 of CTE clock frequency.
21–23 CLKDIV_1	1st Clock divider This value sets the frequency of the 1st clock dividers which will be used to generate the clock type signal. **NOTE:** This value is taken in power of 2 and the divided clock is of 50% duty cycle. 000 First clock divider's frequency is equal to CTE clock frequency. 001 First clock divider's frequency is equal to half the CTE clock frequency. 111 First clock divider's frequency is equal to 1/128 of the CTE clock frequency.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 CTECK_DV	CTE Clock divider These 6 bits are used to define the CTE datapth clock. The value ranges from 0-63 where 0 and 1 mean the divide ratio is 1. CTE datapath runs on this clock. **NOTE:** Value 0 and 1 will result in the same clock frequency as main CTE clock and rest of the values will result in the divided clock e.g. if it is 2 then a divide by 2 clock will be resulted. Duty cycle is not equal to 50%. 000000 to 000001 Datapath is operating at CTE clock frequency. 000010 Datapath is operating at half the CTE clock frequency. 000011 Datapath is operating at 1/3 of CTE clock frequency. 111111 Datapath is operating at 1/63 of CTE clock frequency.

46.7.3 First Timing Table Register (LSB) (CTE_LUT0_LSB*n*)

These 32 registers are used to implement the first timing table which is used to configure the signal state at particular timing instances.

Address: 0h base + 8h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SPT_EVT				SPT_PHIV	DAC_ADE	WG_LSEL		WG_CLD	WG_RST	WG_TGM	WG_UDM	WG_HLD	WG_RUN	RCV_OVF	ADC_OVF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIME_0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_LUT0_LSB*n* field descriptions

Field	Description
0–3 SPT_EVT	SPT Event These signals are used to direct the command execution in SPT i.e. using these in a sequence helps debugging the SPT command execution sequence.
4 SPT_PHIV	SPT phase invert. This signal is sent to the SPT block to invert the phase of the signal before processing it. 0 Normal SPT processing requested. 1 SPT should invert the phase of the sample before processing.
5 DAC_ADE	DAC active data edge The intention of DAC_ADE feature is to switch the clock phase of the DAC output between radar chirps. With ADE==0, the WGM sends the first DAC word at the rising edge of the WGM clock = rising edge of dac clock and then it is stable till the next DAC clock capture edge. With ADE==1, the WGM sends the first DAC word at the rising edge of the next WGM clock = falling edge of dac clock and then it is stable till the next DAC clock capture edge.
6–7 WG_LSEL	WGM LUT select This field is sent to the WGM block to select among the 4 available LUTs in order to send data towards the DAC. 00 LUT0 is selected. 01 LUT1 is selected. 10 LUT2 is selected. 11 LUT3 is selected.

Table continues on the next page...

CTE_LUT0_LSB*n* field descriptions (continued)

Field	Description
8 WG_CLD	WGM counter load This bit when asserted loads the WGM address counter with CVAL on the next WGM clock. 0 De-asserted. 1 WGM address counter is pre-loaded with CVAL.
9 WG_RST	WGM reset This bit when asserted resets the WGM address counter to 0 on the next WGM clock. 0 Deasserted. 1 WGM address counter is reset.
10 WG_TGM	WGM toggle mode This bit when asserted forces the address counter in toggle mode. In toggle mode, counter first increments and decrements in the next WGM clock. 0 Toggle mode disabled. 1 Toggle mode enabled.
11 WG_UDM	WGM up/down mode This bit decides the mode in which the address counter operates. 0 Decrement mode (default). 1 Increment mode.
12 WG_HLD	WGM hold This signal when asserted replaces the WGM address which is used to read the LUT by FADR asynchronously. See the WGM block diagram for more details. 0 De-asserted. 1 WGM goes to the HALT mode.
13 WG_RUN	WGM run This signal is sent to the WGM module and is described below. 0 De-asserted. 1 Signal is asserted meaning that the WGM address counter is running based on wg_udm/wg_tgm signal states.
14 RCV_OVF	Receiver Overflow Mask This signal is used to enable the counter which counts the duration of the receive overflow counter at offset 0x019c. 0 Receiver overflow event is masked and counter is disabled. 1 Receiver overflow event is not masked and counter is enabled.
15 ADC_OVF	ADC overflow mask This signal is used to enable the counter which counts the duration of the ADC overflow counter at offset 0x0198.

Table continues on the next page...

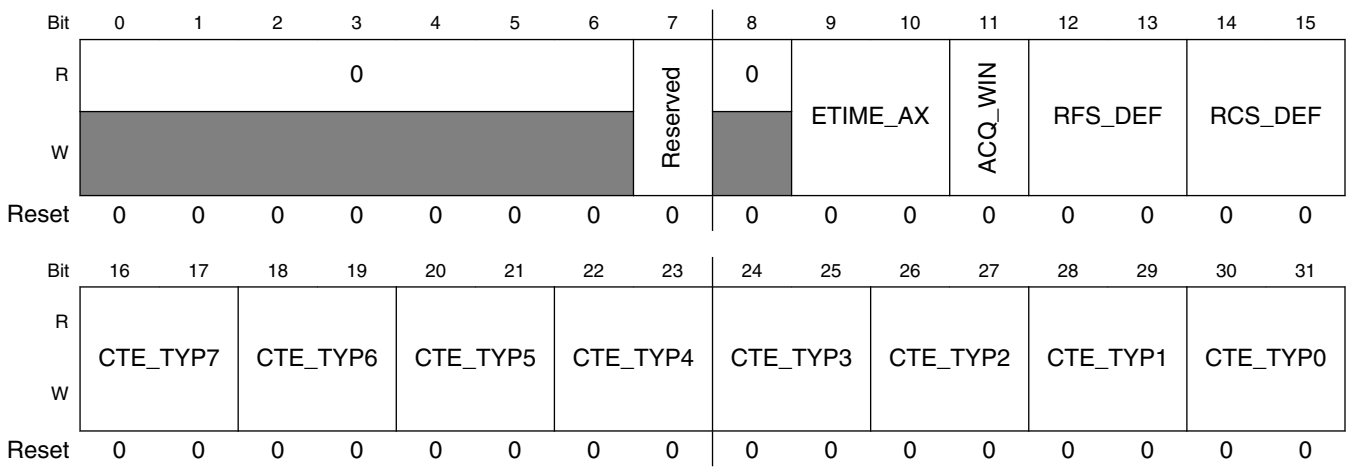
CTE_LUT0_LSB*n* field descriptions (continued)

Field	Description
	0 ADC overflow event is masked and counter is disabled. 1 ADC overflow event is not masked and counter is enabled.
16–31 TIME_0	Time instance This 16-bit field is used to enter the time instance for a particular signal definition. All the signals in the signal definition fields are associated with this particular instance of time. **NOTE:** This field should be programmed with a non-zero value if the table entry is valid and will be used to provide signal states.

46.7.4 First Timing Table Register (MSB) (CTE_LUT0_MSB*n*)

These 32 registers are used to define the signal state of the remaining signals which were not accommodated in LUT_LSB_0 register.

Address: 0h base + 88h offset + (4d × i), where i=0d to 31d



CTE_LUT0_MSB*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–10 ETIME_AX	ETimer Auxiliary These bits are going to the ETimer in the system. 00 Both the eTimer_AUX_0 and eTimer_AUX_1 are DISABLED. 01 eTimer_AUX_0 is ENABLED and eTimer_AUX_1 is DISABLED.

Table continues on the next page...

CTE_LUT0_MSB*n* field descriptions (continued)

Field	Description
	10 eTimer_AUX_0 is DISABLED and eTimer_AUX_1 is ENABLED. 11 Both the eTimer_AUX_0 and eTimer_AUX_1 are ENABLED.
11 ACQ_WIN	CTE Acquisition Window This signal is sent to the SPT block for starting the acquisition process. 0 Acquisition process in SPT has not started. 1 Acquisition process in SPT has started.
12–13 RFS_DEF	RFS definition These bits define the signal state of the RFS signal if the CTE is operating in master mode. Signal behavior is same as that of ctep0 defined below.
14–15 RCS_DEF	RCS definition These bits define the signal state of the RCS signal if the CTE is operating in master mode. Signal behavior is same as that of ctep0 defined below.
16–17 CTE_TYP7	CTE Pulses These bits define the signal state of the 8th bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
18–19 CTE_TYP6	CTE Pulses These bits define the signal state of the 7th bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
20–21 CTE_TYP5	CTE Pulses These bits define the signal state of the 6th bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
22–23 CTE_TYP4	CTE Pulses These bits define the signal state of the 5th bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
24–25 CTE_TYP3	CTE Pulses These bits define the signal state of the 4th bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
26–27 CTE_TYP2	CTE Pulses These bits define the signal state of the 3rd bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
28–29 CTE_TYP1	CTE Pulses These bits define the signal state of the 2nd bit of the CTEP bus. Signal behavior is same as that of ctep0 defined in CTE_LUT0_MSB.
30–31 CTE_TYP0	CTE Pulses These bits define the signal state of the 1st bit of the CTEP bus. **NOTE:** Please note that when CTEP signals are defined as clock type, the frequency of the CTEP signal is programmable using the CTE_CLKSEL register. In this case, CTE_CLKSEL setting needs to be defined in accordance with the PAD characteristics. Please refer IO PAD characteristics in datasheet for details.

Table continues on the next page...

CTE_LUT0_MSB*n* field descriptions (continued)

Field	Description
	00 ctep0 signal will be driven with '0' irrespective of the signal type. 01 ctep0 signal will be driven with '1' in case of logic type and toggle type. In case of clock type, divided clock with a synchronous rising edge will be sent. See CNTRL1 for divided clock description. 10 ctep0 signal will be driven with 'Z' in case of signal type is logic. ctep0 will toggle its value toggle in case of signal type is toggle. Free running clock output of one of the 4 clock dividers will be sent on ctep0 if the type is defined as clock. 11 No change in the signal level in case of signal type is logic and toggle. ctep0 signal will be driven with '1' in case of signal type is clock.

46.7.5 Second Timing Table Register (LSB) (CTE_LUT1_LSB*n*)

These 32 registers are used to implement the second timing table which is used to configure the signal state at particular timing instances.

Address: 0h base + 108h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SPT_EVT				SPT_PHIV	DAC_ADE	WG_LSEL		WG_CLD	WG_RST	WG_TGM	WG_UDM	WG_HLD	WG_RUN	RCV_OVF	ADC_OVF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIME_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_LUT1_LSB*n* field descriptions

Field	Description
0–3 SPT_EVT	SPT Event These signals are used to suspend the command execution in SPT i.e. using these in a sequence helps debugging the SPT command execution sequence.
4 SPT_PHIV	SPT phase invert. This signal is sent to the SPT block to invert the phase of the signal before processing it. 0 Normal SPT processing requested. 1 SPT should invert the phase of the sample before processing.
5 DAC_ADE	DAC active data edge The intention of DAC_ADE feature is to switch the clock phase of the DAC output between radar chirps.

Table continues on the next page...

CTE_LUT1_LSB*n* field descriptions (continued)

Field	Description
	With ADE==0, the WGM sends the first DAC word at the rising edge of the WGM clock = rising edge of dac clock and then it is stable till the next DAC clock capture edge. With ADE==1, the WGM sends the first DAC word at the rising edge of the next WGM clock = falling edge of dac clock and then it is stable till the next DAC clock capture edge.
6–7 WG_LSEL	WGM LUT select This field is sent to the WGM block to select among the 4 available LUTs. 00 LUT0 is selected. 01 LUT1 is selected. 10 LUT2 is selected. 11 LUT3 is selected.
8 WG_CLD	WGM counter load This bit when asserted loads the WGM address counter with CVAL. 0 De-asserted. 1 WGM address counter is pre-loaded with CVAL.
9 WG_RST	WGM reset This bit when asserted resets the WGM address counter to 0. 0 Deasserted. 1 WGM address counter is reset.
10 WG_TGM	WGM toggle mode This bit when asserted force the address counter in toggle mode. In toggle mode, counter first increments and decrements in the next cycle. 0 Toggle mode disabled. 1 Toggle mode enabled.
11 WG_UDM	WGM up/down mode This bit decides the mode in which the address counter operates. 0 Decrement mode (default). 1 Increment mode.
12 WG_HLD	WGM hold This signal is used by the WGM address counter FSM. 0 De-asserted. 1 Address FSM goes to the HALT state.
13 WG_RUN	WGM run This signal is sent to the WGM module and is described below: 0 De-asserted. 1 Signal is asserted meaning that the WGM address counter FSM starts.
14 RCV_OVF	Receiver Overflow Mask

Table continues on the next page...

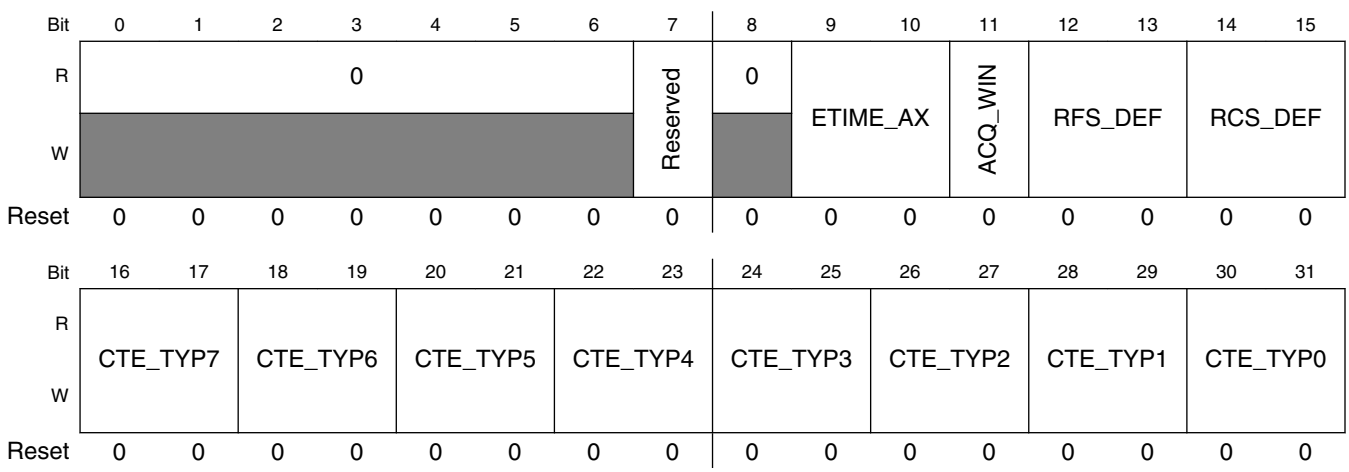
CTE_LUT1_LSB*n* field descriptions (continued)

Field	Description
	This signal is used to enable the counter which counts the duration of the receive overflow counter at offset 0x019c. 0 Receiver overflow event is not masked and counter is enabled. 1 Receiver overflow event is masked and counter is disabled.
15 ADC_OVF	ADC overflow mask This signal is used to enable the counter which counts the duration of the ADC overflow counter at offset 0x0198. 0 Receiver overflow event is masked and counter is disabled. 1 Receiver overflow event is not masked and counter is enabled.
16–31 TIME_1	Time instance This 16 bit field is used to enter the time instance for a particular signal definition. All the signals in the signal definition fields are associated with this particular instance of time. **NOTE:** This field should be programmed with a non-zero value if the table entry is valid and will be used to provide signal states.

46.7.6 Second Timing Table Register (MSB) (CTE_LUT1_MSB*n*)

This 32 register are used to define the signal behaviors of the remaining signals which were not accommodated in LUT_LSB_1

Address: 0h base + 188h offset + (4d × i), where i=0d to 31d



CTE_LUT1_MSB*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CTE_LUT1_MSB*n* field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–10 ETIME_AX	ETimer Auxiliary These bits are going to the ETimer in the system. 00 Both the eTimer_AUX_0 and eTimer_AUX_1 are DISABLED. 01 eTimer_AUX_0 is ENABLED and eTimer_AUX_1 is DISABLED. 10 eTimer_AUX_0 is DISABLED and eTimer_AUX_1 is ENABLED. 11 Both the eTimer_AUX_0 and eTimer_AUX_1 are ENABLED.
11 ACQ_WIN	CTE Acquisition Window This signal is sent to the SPT block for starting the acquisition process. 0 Acquisition process in SPT has not started. 1 Acquisition process in SPT has started.
12–13 RFS_DEF	RFS definition These bits define the signal state of the RFS signal if the CTE is operating in master mode. Bit map description is same as ctep signal defined in LUT0_MSB.
14–15 RCS_DEF	RCS definition These bits define the signal state of the RCS signal if the CTE is operating in master mode. Bit map description is same as ctep signal defined in LUT0_MSB.
16–17 CTE_TYP7	CTE Pulses These bits define the signal state of the 8th bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
18–19 CTE_TYP6	CTE Pulses These bits define the signal state of the 7th bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
20–21 CTE_TYP5	CTE Pulses These bits define the signal state of the 6th bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
22–23 CTE_TYP4	CTE Pulses These bits define the signal state of the 5th bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
24–25 CTE_TYP3	CTE Pulses These bits define the signal state of the 4th bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
26–27 CTE_TYP2	CTE Pulses These bits define the signal state of the 3rd bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.

Table continues on the next page...

CTE_LUT1_MSB*n* field descriptions (continued)

Field	Description
28–29 CTE_TYP1	CTE Pulses These bits define the signal state of the 2nd bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB.
30–31 CTE_TYP0	CTE Pulses These bits define the signal state of the 1st bit of the CTEP bus. Bit map description is same as ctep signals defined in LUT0_MSB. **NOTE:** When CTEP signals are defined as clock type, the frequency of clock is programmable using the CTE_CLKSEL register but the actual frequency is limited by the PAD characteristics. Please refer datasheet for details.

46.7.7 Signal Type Register 0 (CTE_SIGTYPE0*n*)

These two register are used to define the type of various internal and external signals corresponding to the timing tables. For example, if the signal is clock type or pulse type or logic type. SIGTYPE0_0 corresponds to TT0 and SIGTYPE0_1 corresponds to TT1.

Address: 0h base + 208h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CTE_TYP7		CTE_TYP6		CTE_TYP5		CTE_TYP4		CTE_TYP3		CTE_TYP2		CTE_TYP1		CTE_TYP0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	STP_EVT				PH_INVT	DAC_ADE	WG_LSEL	WG_CLD	WG_RST	WG_TGM	WG_UDM	WG_HLD	WG_RUN	RCV_OVF	ADC_OVF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_SIGTYPE0*n* field descriptions

Field	Description
0–1 CTE_TYP7	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.

Table continues on the next page...

CTE_SIGTYPE0*n* field descriptions (continued)

Field	Description
2–3 CTE_TYP6	CTE pulse type These 2 bits define the signal type of this signal as follows. 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
4–5 CTE_TYP5	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
6–7 CTE_TYP4	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
8–9 CTE_TYP3	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
10–11 CTE_TYP2	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
12–13 CTE_TYP1	CTE pulse type These 2 bits define the signal type of this signal as follows: 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
14–15 CTE_TYP0	CTE pulse type These 2 bits define the signal type of this signal as follows:

Table continues on the next page...

CTE_SIGTYPE0*n* field descriptions (continued)

Field	Description
	00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–20 STP_EVT	SPT event type These bits defines the signal type as follows for all the 4 bits:. 0 Signal type is logic. 1 Signal type is pulse
21 PH_INVT	Phase invert type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
22 DAC_ADE	DAC active data edge type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
23 WG_LSEL	WGM LUT select type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
24 WG_CLD	WGM counter load type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
25 WG_RST	WGM reset type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
26 WG_TGM	WGM toggle mode type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
27 WG_UDM	WGM up/down mode type This bit defines the signal type as follows:

Table continues on the next page...

CTE_SIGTYPE0*n* field descriptions (continued)

Field	Description
	0 Signal type is logic. 1 Signal type is pulse
28 WG_HLD	WGM hold type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
29 WG_RUN	WGM run type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
30 RCV_OVF	Receive overflow mask type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
31 ADC_OVF	ADC overflow mask type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse

46.7.8 Signal Type Register 1 (CTE_SIGTYPE1*n*)

These 2 register are used to define the type of the remaining timing signals. SIGTYPE1_0 corresponds to TT0 and SIGTYPE1_1 corresponds to TT1.

Address: 0h base + 210h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							Reserved	0	ETIME_AUX		ACQ_WIN	RFS		RCS	
W																
Reset	0	0	0	0	0	0	0	x*	0	0	0	0	0	0	0	0

* Notes:
- This field should not be modified.x = Undefined at reset.

CTE_SIGTYPE1*n* field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–26 ETIME_AUX	Etime Aux Type This bit defines the signal type as follows: 00 Signal Type of eTimer_AUX_0 and eTimer_AUX_1 are Logic. 01 Signal Type of eTimer_AUX_0 is Pulse and eTimer_AUX_1 is Logic. 10 Signal Type of eTimer_AUX_0 is Logic and eTimer_AUX_1 is Pulse. 11 Signal Type of eTimer_AUX_0 and eTimer_AUX_1 are Pulse.
27 ACQ_WIN	Acquisition Window Type This bit defines the signal type as follows: 0 Signal type is logic. 1 Signal type is pulse
28–29 RFS	Radar Frame Synchronization

Table continues on the next page...

CTE_SIGTYPE1*n* field descriptions (continued)

Field	Description
	00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.
30–31 RCS	Radar Chirp Synchronization 00 Signal state will be 'Z'. 01 Signal type is Toggle. 10 Signal type is Clock. 11 Signal type is Logic.

46.7.9 CTE Interrupt Enable Register (CTE_INTEN)

This is the interrupt enable register for CTE interrupts. An interrupt will be generated if the enable bit is set and the corresponding event in the interrupt status register occurs.

NOTE

Please note that in case of continuous toggle mode (CNTRL[OPMOD_SL] = "11"), if user enables both start and end interrupts for both the timing tables (TT0_STEN, TT1_STEN, TT0_EDEN and TT1_EDEN), it might result into a scenario where CPU is serving interrupt very frequently. The user should judiciously enable the appropriate interrupts to avoid such scenarios.

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						LST_EXEN	eDMA_TEN	RFS_EN	RCS_EN	0	AC_WD_EN	TT1_EDEN	TT0_EDEN	TT1_STEN	TT0_STEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

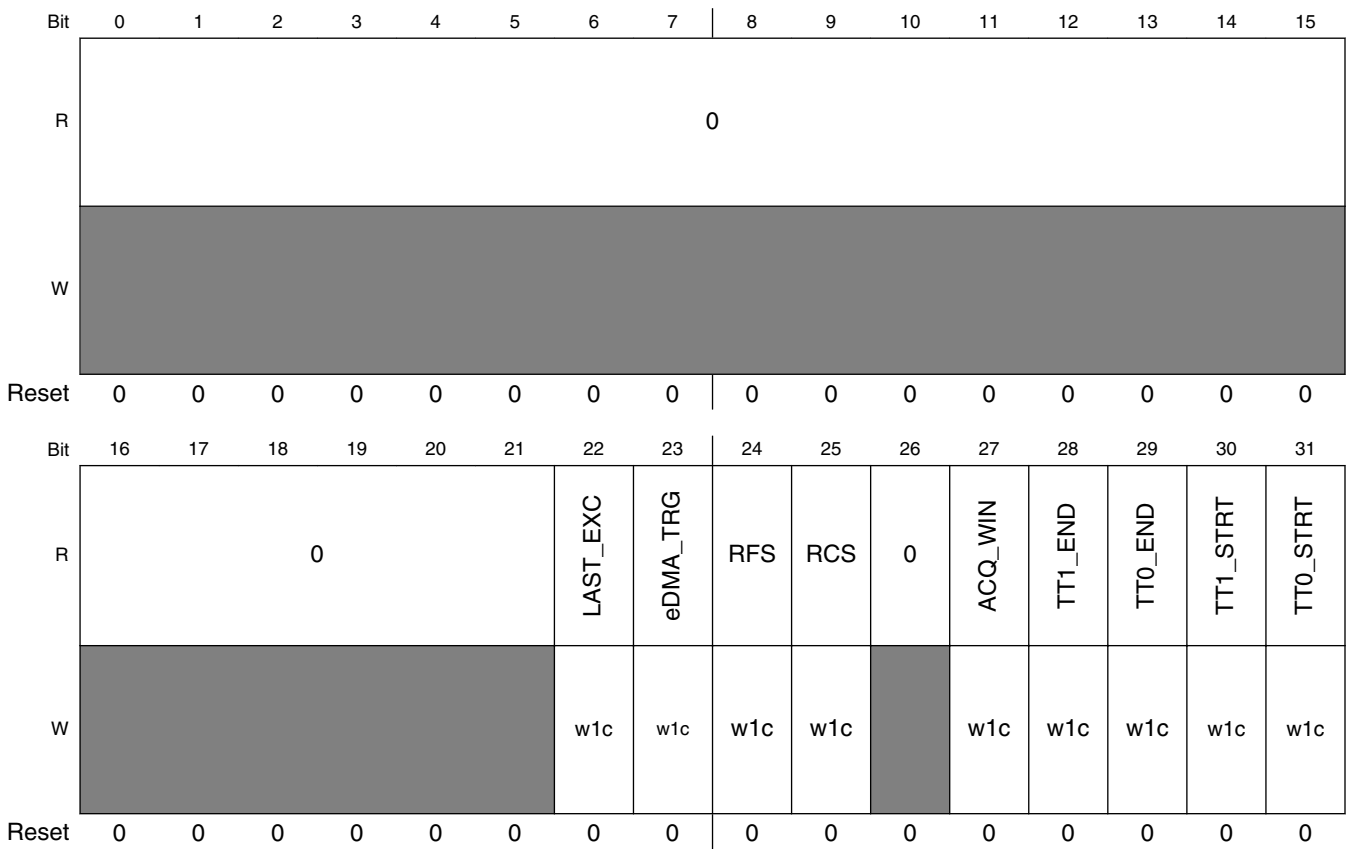
CTE_INTEN field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LST_EXEN	Last Table Execution Enable 0 Disabled. 1 An interrupt will be generated when the execution of the timing table is finished for the final time as configured.
23 eDMA_TEN	eDMA Trigger Interrupt Enable 0 Disabled 1 An interrupt will be generated on the rising edge of the eDMA trigger.
24 RFS_EN	RFS interrupt enable (rising edge) 0 Disabled. 1 An interrupt will be generated on the rising edge of the RFS.
25 RCS_EN	RCS interrupt enable (rising edge) 0 Disabled. 1 An interrupt will be generated on the rising edge of the RCS.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 AC_WD_EN	Acquisition window (rising) 0 Disabled. 1 An interrupt will be generated on the rising edge of the acquisition window signal.
28 TT1_EDEN	Timing table end (rising) 0 Disabled. 1 An interrupt will be generated on the rising edge of the second timing table end signal.
29 TT0_EDEN	Timing table end (rising) 0 Disabled. 1 An interrupt will be generated on the rising edge of the first timing table end signal.
30 TT1_STEN	Timing table start (rising) 0 Disabled. 1 An interrupt will be generated on the rising edge of the second timing table start signal.
31 TT0_STEN	Timing table start (rising) 0 Disabled. 1 An interrupt will be generated on the rising edge of the first timing table start signal.

46.7.10 CTE Interrupt Status Register (CTE_INTSTAT)

This is the interrupt status register which contains flags that are set when events occur that generate interrupts, if the corresponding mask bit is set. These bits are set on the events and can be cleared by w1c strategy. If an event is in CTE clock domain, it will take 1 CTE clock + 3 IPG clock cycles for the corresponding status bit in this register to be set. In case of CTE running in divided clock, the timing taken will be 1 divided CTE clock + 3 IPG clock cycles. For a signal which is in IPG clock domain, it will take 2 IPG clock cycles for the corresponding event to be set.

Address: 0h base + 224h offset = 224h



CTE_INTSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LAST_EXC	Last Execution This bit is set and remains set until w1c when the timing table is executed for the last time as per the configuration. For example if CNTRL[REP_CNT] is 5 and CNTRL[OPMOD_SL] = "10", this bit will be set

Table continues on the next page...

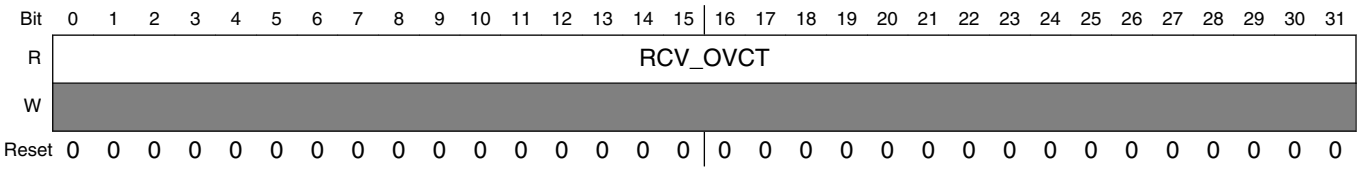
CTE_INTSTAT field descriptions (continued)

Field	Description
	when the last entry of the TT0 is executed for the 5th time. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
23 eDMA_TRG	eDMA Trigger This bit is set and remains set until w1c when a rising edge of the eDMA trigger signal is detected. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
24 RFS	Radar Frame Synchronization This bit is set and remains set until w1c when a rising edge of the RFS signal is detected. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
25 RCS	Radar Chirp Synchronization This bit is set and remains set until w1c when a rising edge of the RCS signal is detected. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 ACQ_WIN	Acquisition window (rising) This bit is set and remains set until w1c when acquisition signal from CTE to the SPT is asserted. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
28 TT1_END	TT1 end This bit is set and remains set until w1c when TT1 execution reaches the end. **NOTE:** The end of table is generated when the entry address counter reaches the last valid entry. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
29 TT0_END	TT0 end This bit is set and remains set until w1c when TT0 execution reaches the end. **NOTE:** The end of table will be generated when the entry address counter reaches the last valid entry. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
30 TT1_STRT	TT1 start This bit is set and remains set until w1c when timing table execution for TT1 is started. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.
31 TT0_STRT	TT0 start This bit is set and remains set until w1c when timing table execution for TT0 is started. An interrupt is generated when the corresponding enable bit is set in the interrupt enable register.

46.7.11 Receiver Overflow Counter (CTE_RCVOFCNT)

This register contains the Receiver overflow event counter.

Address: 0h base + 22Ch offset = 22Ch



CTE_RCVOFCNT field descriptions

Field	Description
0–31 RCV_OVCT	Receiver Overflow counter This counter increments on every CTE clock when the RCV overflow mask bit in the timing table is set and there is an receiver overflow event at the input. **NOTE:** Since this counter is running in CTE clock domain, read to this will be associated with the wait cycles caused because of the asynchronous nature of CTE and IPG clocks.

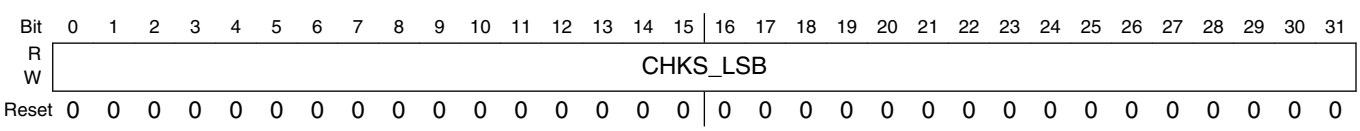
46.7.12 LUT Checksum Register (CTE_CKSM_LSB)

The following register contains the checksum value for the signal definition bits in the CTE_LUT0/CTE_LUT1_LSB and CTE_LUT0/CTE_LUT1_MSB registers. Please refer to Checksum for LUT0/LUT1 section for details of Checksum.

NOTE

A portion of 40:0 is reflected back on read of this register.

Address: 0h base + 270h offset = 270h



CTE_CKSM_LSB field descriptions

Field	Description
0–31 CHKS_LSB	Checksum Result LSB This value is the result of the checksum calculation for 16 timing signals in the LUT0/LUT1_LSB register ([31:16]) and 16 timing signals in LUT0/LUT1_MSB register([15:0]). The checksum result is updated every time these timing signals are updated.

46.7.13 LUT Checksum Register (CTE_CKSM_MSB)

Following register contains the checksum value for the remaining signal definition bits the LUT0/LUT1_MSB register. Please refer to Checksum for LUT0/LUT1 section for details of Checksum.

NOTE

A portion of 40:0 is reflected back on read of this register.

Address: 0h base + 274h offset = 274h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							0	CHKS_MSB							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

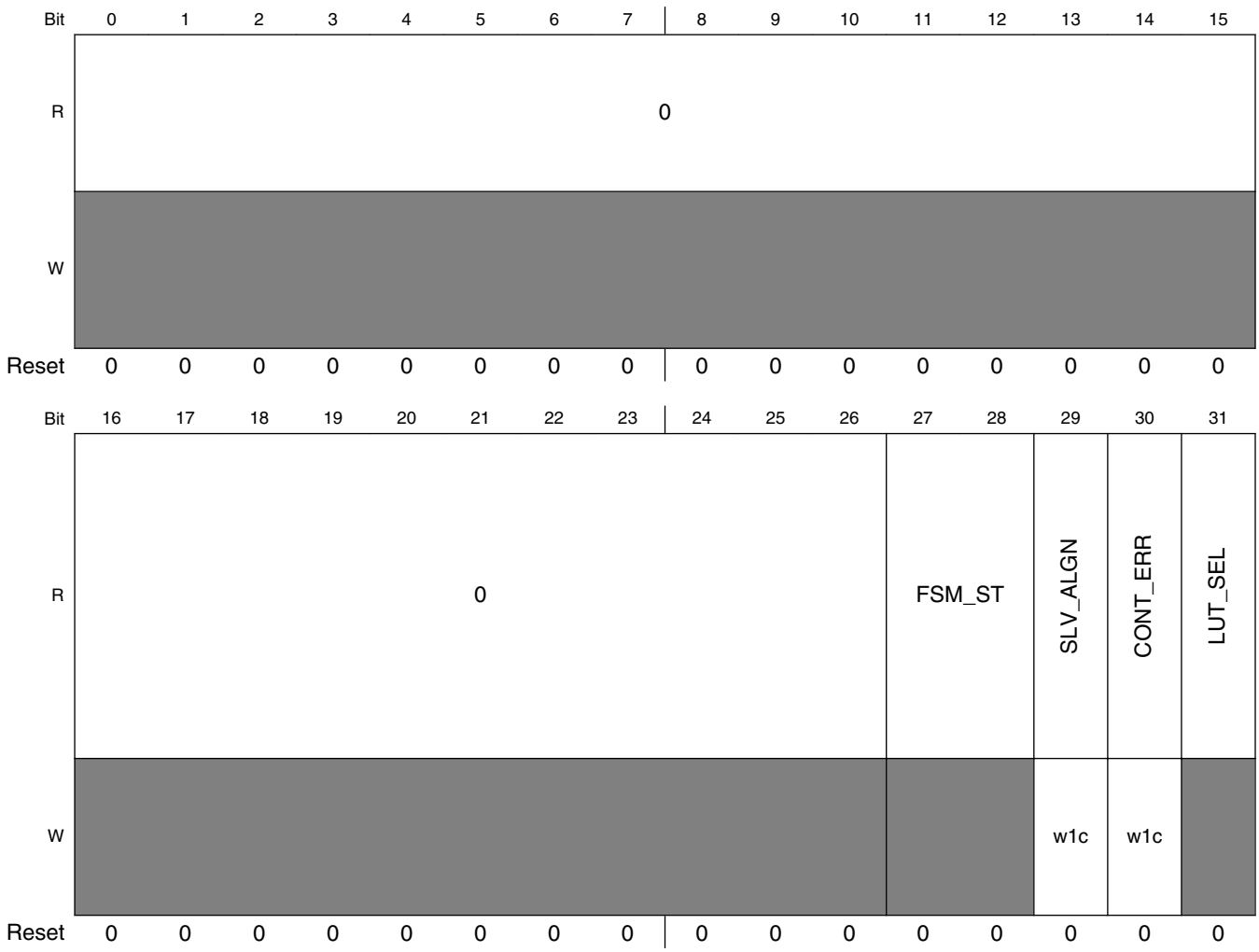
CTE_CKSM_MSB field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CHKS_MSB	Checksum Result LSB This value is the result of the checksum calculation for 8 timing signals in the LUT0/LUT1_MSB[24:16]. The checksum result is updated every time these timing signals are updated.

46.7.14 Debug Register (CTE_DBG_REG)

Following register is used for the debugging purpose.

Address: 0h base + 278h offset = 278h



CTE_DBG_REG field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–28 FSM_ST	OPMOD FSM state These bits reflect the state of the operation mode FSM. These can be used for debugging purpose. 00 FSM is in HALT. 01 FSM is in CRUN Table 0 10 FSM is in CRUN Table 1

Table continues on the next page...

CTE_DBG_REG field descriptions (continued)

Field	Description
29 SLV_ALGN	RFS/RCS align status This bit is set when the CTE is configured to work in slave mode (CNTRL[MA_SL_ST] is set) and first RCS pulse delayed by CTE_CNTRL[RCS_DLY] value, is received. This bit is valid in slave mode only. This bit will be reset using w1c strategy. **NOTE:** The RCS pulse is the synchronized and delayed i.e. the input RCS pulse is first synchronized using a 2 flop mechanism and then fed to the delay block which will delay the pulse as per configured value of CNTRL[RCS_DLY]. 0 RFS and RCS input are not aligned. 1 RFS and RCS input are aligned.
30 CONT_ERR	Timing Table contention error This bit is set when a timing table, which is currently active, is attempted to be programmed/written by CPU, eDMA, or by any other means. It is cleared by writing a 1 on it. 0 No timing table contention has occurred. 1 Contention has occurred on the active timing table.
31 LUT_SEL	TT Select This bit reflects which timing table out of the 2 available is currently selected. This field is read only, it will be toggling when toggle mode (CTE_CNTRL[OPMOD_SL] is "11") is selected. 0 TT0 is currently being used for signal definitions. 1 TT1 is currently being used for signal definitions.

46.7.15 TT0 Execution Duration Register (CTE_LUT_DUR)

This register contains the counter value which dictates for how many clock cycles the active TT0 is executed.

NOTE

In Toggle mode, if one of the CTE LUT duration registers (CTE_LUT_DUR or CTE_LUT_DUR1) is programmed zero and other is programmed less than 32, then the CTE timing table executes all 32 entries in for *both* LUTs.

Specifically, in Toggle mode (CNTRL[OPMOD_SL] =="2'b11"), when both LUTs are running, and if either the CTE_LUT_DUR or the CTE_LUT_DUR1 register is programmed "0", then following behavior is expected:

- If CTE_LUT_DUR == "0" and CTE_LUT_DUR1 != "0", then both LUTs are executed in full. A total of 64 entries are executed (32 for LUT0 and 32 for LUT1).
- If CTE_LUT_DUR != "0" and CTE_LUT_DUR1 == "0", then:
  - Only on the first run only, LUT0 is executed to programmed delay as specified by CTE_LUT_DUR.
  - On all further runs, all 32 entries are executed. A total of 64 entries are executed (32 for LUT0 and 32 for LUT1).

To run the 32 entries for the intended LUT, the user must not program either of the CTE LUT duration registers to “0”. The user has to program the correct value to *both* the CTE_LUT_DUR or CTE_LUT_DUR1 registers.

Address: 0h base + 27Ch offset = 27Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TT0_DUR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_LUT_DUR field descriptions

Field	Description
0–31 TT0_DUR	TT0 Duration This counter defines the duration in terms of the CTE clock cycles for which the first timing table is executed. It starts decrementing as soon as the timing table execution is started and once it reaches 0, timing table end is considered. If it is 0, all the 32 table entries in TT0 will be executed before declaring the end of table.

46.7.16 TT1 Execution Duration Register (CTE_LUT_DUR1)

This register contains the counter value which dictates for how many clock cycles the active TT1 is executed.

NOTE

In Toggle mode, if one of the CTE LUT duration registers (CTE_LUT_DUR or CTE_LUT_DUR1) is programmed zero and other is programmed less than 32, then the CTE timing table executes all 32 entries in for *both* LUTs.

Specifically, in Toggle mode (CNTRL[OPMOD_SL] =="2'b11"), when both LUTs are running, and if either the CTE_LUT_DUR or the CTE_LUT_DUR1 register is programmed "0", then following behavior is expected:

- If CTE_LUT_DUR == "0" and CTE_LUT_DUR1 != "0", then both LUTs are executed in full. A total of 64 entries are executed (32 for LUT0 and 32 for LUT1).
- If CTE_LUT_DUR != "0" and CTE_LUT_DUR1 == "0", then:
  - Only on the first run only, LUT0 is executed to programmed delay as specified by CTE_LUT_DUR.
  - On all further runs, all 32 entries are executed. A total of 64 entries are executed (32 for LUT0 and 32 for LUT1).

To run the 32 entries for the intended LUT, the user must not program either of the CTE LUT duration registers to “0”. The user has to program the correct value to *both* the CTE_LUT_DUR or CTE_LUT_DUR1 registers.

Address: 0h base + 280h offset = 280h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TT1_DUR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_LUT_DUR1 field descriptions

Field	Description
0–31 TT1_DUR	TT1_DURATION This register contains the counter value which dictates for how many CTE clock cycles the TT1 is executed. If it is 0, all 32 entries in TT1 will be executed.

46.7.17 Clock Select Register (CTE_CLKSEL)

Following register is used to select which one of the 4 independent clock dividers is used to send the clock on CTEP[7:0], RCS and RFS signals when these are defined as clock type.

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CLK_SEL9		CLK_SEL8		CLK_SEL7		CLK_SEL6		CLK_SEL5		CLK_SEL4		CLK_SEL3		CLK_SEL2		CLK_SEL1		CLK_SEL0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTE_CLKSEL field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–13 CLK_SEL9	Clock Select These bits are used to select one of the 4 available clock dividers for RFS when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on RFS. 01 2nd clock divider will be used for generating the clock type signal on RFS. 10 3rd clock divider will be used for generating the clock type signal on RFS. 11 4th clock divider will be used for generating the clock type signal on RFS.
14–15 CLK_SEL8	Clock Select These bits are used to select one of the 4 available clock dividers for RCS when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on RCS. 01 2nd clock divider will be used for generating the clock type signal on RCS. 10 3rd clock divider will be used for generating the clock type signal on RCS. 11 4th clock divider will be used for generating the clock type signal on RCS.
16–17 CLK_SEL7	Clock Select These bits are used to select one of the 4 available clock dividers for ctep7 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep7. 01 2nd clock divider will be used for generating the clock type signal on ctep7. 10 3rd clock divider will be used for generating the clock type signal on ctep7. 11 4th clock divider will be used for generating the clock type signal on ctep7.
18–19 CLK_SEL6	Clock Select These bits are used to select one of the 4 available clock dividers for ctep6 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep6. 01 2nd clock divider will be used for generating the clock type signal on ctep6.

Table continues on the next page...

CTE_CLKSEL field descriptions (continued)

Field	Description
	10 3rd clock divider will be used for generating the clock type signal on ctep6. 11 4th clock divider will be used for generating the clock type signal on ctep6.
20–21 CLK_SEL5	Clock Select These bits are used to select one of the 4 available clock dividers for ctep5 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep5. 01 2nd clock divider will be used for generating the clock type signal on ctep5. 10 3rd clock divider will be used for generating the clock type signal on ctep5. 11 4th clock divider will be used for generating the clock type signal on ctep5.
22–23 CLK_SEL4	Clock Select These bits are used to select one of the 4 available clock dividers for ctep4 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep4. 01 2nd clock divider will be used for generating the clock type signal on ctep4. 10 3rd clock divider will be used for generating the clock type signal on ctep4. 11 4th clock divider will be used for generating the clock type signal on ctep4.
24–25 CLK_SEL3	Clock Select These bits are used to select one of the 4 available clock dividers for ctep3 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep3. 01 2nd clock divider will be used for generating the clock type signal on ctep3. 10 3rd clock divider will be used for generating the clock type signal on ctep3. 11 4th clock divider will be used for generating the clock type signal on ctep3.
26–27 CLK_SEL2	Clock Select These bits are used to select one of the 4 available clock dividers for ctep2 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep2. 01 2nd clock divider will be used for generating the clock type signal on ctep2. 10 3rd clock divider will be used for generating the clock type signal on ctep2. 11 4th clock divider will be used for generating the clock type signal on ctep2.
28–29 CLK_SEL1	Clock Select. These bits are used to select one of the 4 available clock dividers for ctep1 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep1. 01 2nd clock divider will be used for generating the clock type signal on ctep1. 10 3rd clock divider will be used for generating the clock type signal on ctep1. 11 4th clock divider will be used for generating the clock type signal on ctep1.
30–31 CLK_SEL0	Clock Select These bits are used to select one of the 4 available clock dividers for ctep0 when it is defined as clock type. 00 1st clock divider will be used for generating the clock type signal on ctep0.

Table continues on the next page...

CTE_CLKSEL field descriptions (continued)

Field	Description
	01 2nd clock divider will be used for generating the clock type signal on ctep0. 10 3rd clock divider will be used for generating the clock type signal on ctep0. 11 4th clock divider will be used for generating the clock type signal on ctep0.

46.8 Functional description

46.8.1 Timing Table (TT) execution

Once a timing table is configured, it is executed by the CTE to generate the required signals. A Finite State Machine (FSM) is responsible for the execution. The FSM has the following states.

- Halt
- Continuous run, single table
- Continuous toggle, two tables

For details, see the following figure.

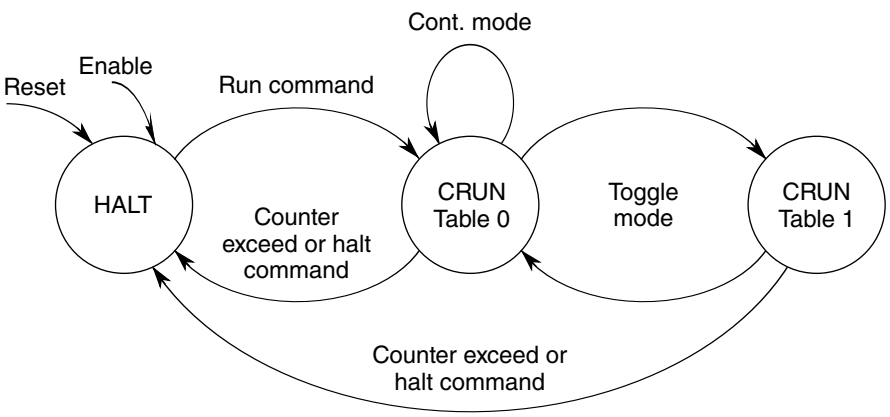


Figure 46-10. Operation mode state diagram

NOTE

Even if the FSM goes back to halt state after counter exceeds, but the CTE_CNTRL[OPMOD_SL] bits will remain as configured by software. Software can read the CTE_DBG_REG[FSM_ST] bits to know the state of the FSM.

In Halt state, timing generation is stopped. All signals are in their default level. HALT state is exited as soon as the CNTRL[CTE_RST] is de-asserted, CNTRL[OPMOD_SL] is set to either 01/10/11, and CNTRL1[CTE_EN] is set.

In CRUNTable0 and CRUNTable1 states, the timing tables are executed for a predefined number of repetitions or forever until the halt state is enforced depending upon the value of CNTRL[REP_CNT]. The maximum count of repetitions is 2^16-1, where the value of 0 is used to repeat forever.

End of timing table is considered either after the last timing entry is reached or when timing table execution duration counter LUT_DUR/LUT_DUR1 expires. In case when the table is executed only once, the signal states will retain their states as according to the last entry of the table which is decided by the timing table execution duration counter (LUT_DUR/LUT_DUR1). If table execution is configured for more than one execution, after reaching the last entry in the timing table, entry address will roll back to 0 driving the signal states as per timing entry 0.

Once the timing table reaches the end, the next timing table is activated (toggle mode) or the same timing table runs again depending on the CNTRL[OPMOD_SL] setting. In toggle mode, an automatic update by the eDMA is possible. The timing tables are toggled either after the last valid timing entry is reached or when LUT_DUR/LUT_DUR1 counter expires. This timing table duration counter is initialized at the beginning of each table along with the time-out value and decrements by 1 on each CTE clock (divided clock if configured).

46.8.2 Timing table

The timing table defines times and signal levels. The times can be defined as absolute intervals from the start of table or relative to previous time entry. Signals generated by CTE can be activated and deactivated with each timing table. The inactive level can be defined separately (0/1/Z). The signal state Z is only valid for signals connected to pads, where pad output is set to high impedance. The time entries define moments, where a change in one of the activated signals happens. It is possible to change the signal mode with each timing table. The last entry in a timing table has at least one clock pulse effect. The supported modes are different for external and internal signals.

46.8.3 Time base

The time base for CTE is derived from its clock. The clock for the CTE is 80MHz. A 6 bit clock divider can be programmed for each timing table using CNTRL[CTECK_DV] in order to modify the scale of time entries.

The divided clocks on which the timing tables are executed are counted with a 16 bit counter, which is compared to the time entries in the table. Once the 16 bit counter matches the time entry value, signal value for all the signals defined in timing table

registers (CTE_LUT0/CTE_LUT1 LSB and CTE_LUT0/CTE_LUT1 MSB) is stored. Based on the type of the signal defined in signal type register (CTE_SIGTYPE0/1), signals at that particular time instant are generated. See the following figure for details:

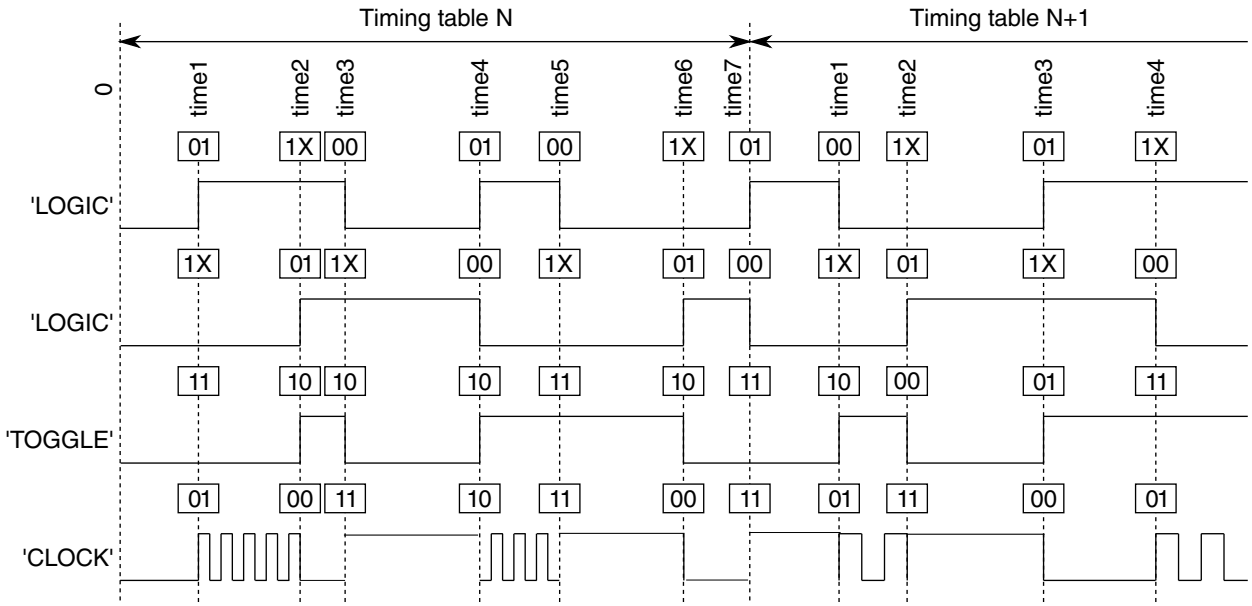


Figure 46-11. Timing generation

46.8.4 Timing generation

The generation of signals is based on configuring timing tables which define the assertion/ de-assertion of signals at specific instants of time. Two timing tables are provided which can be toggled seamlessly without interruption of the timing generation. The signal generation logic works in either the relative mode in which the signal generation logic treats the timing entry in the table as relative to the previous entry or in absolute mode in which each timing entry is treated at absolute time.

46.8.5 CTE clock divider

There are total 5 clock dividers present in CTE module. These are shown in the figure below.

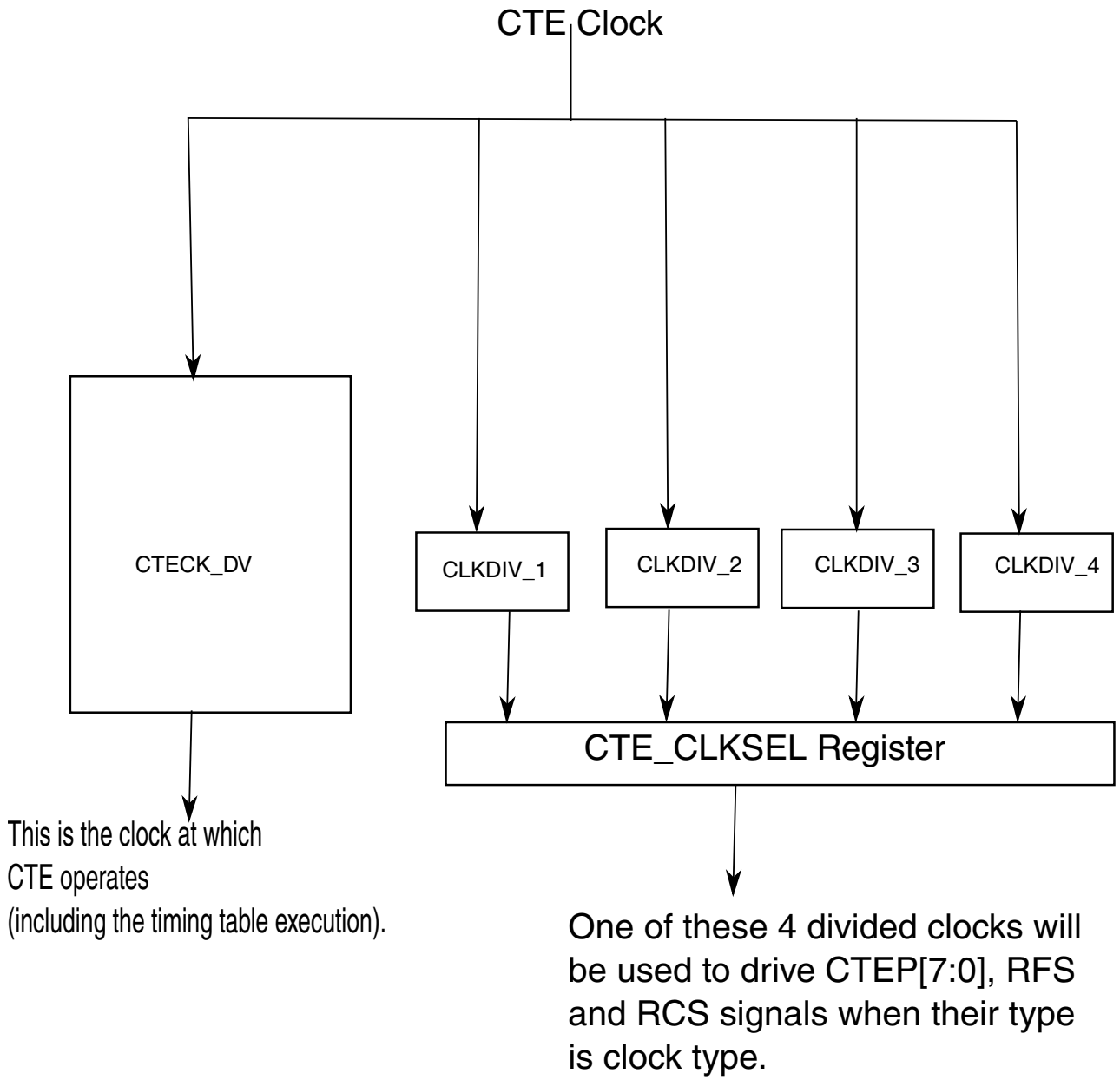


Figure 46-12. CTE Clock divider architecture

The main clock divider derives its frequency from register bits CTE_CNTRL1[CTECK_DV]. The duty cycle of this divided clock is not 50%. The other 4 independent clock dividers derive their frequencies from CTE_CNTRL1[CLKDIV_1], CTE_CNTRL1[CLKDIV_2, CTE_CNTRL1[CLKDIV_3] and CTE_CNTRL1[CLKDIV_4] respectively. The duty cycle of these divided clocks are 50%. These clocks are used to send the periodic signals of defined frequencies on CTEP[7:0], RCS and RFS ports when their type is defined as clock (through CTE signal type0/1 register settings). Please refer the example in initialization sequence to get the details of how user can configure CTE to generate periodic signals on CTEP[7:0], RCS and RFS outputs.

46.8.6 Timing table configuration

The timing tables can be configured either directly by the CPU or through the DMA using the eDMA trigger functionality described below. Note that the timing entries in LUT0_LSB[15:0] and LUT1_LSB[15:0] are the timing instances. The signals will be driven on that particular instance depending upon the relative or absolute mode. Timing entry 0 is not valid and it is the responsibility of the SW that timing entry should not be programmed as 0. If SW enters timing entry as 0, the signal will be driven after 65536 clock cycles because the internal 16 bit timing counter will roll over after these many clock cycles. Similarly in absolute mode, it is the responsibility of the SW to ensure that the next timing entry is always greater than the previous. If this is not maintained, the signal will be driven after 65536+ the value written in the timing entry. See the figure below for details.

First signal output is in absolute mode. The timing entry in the table are 3, 5 and 7 for 3 consecutive locations. Now in absolute mode, the signal state is driven when the internal timecounter reaches the 3rd, 5th and 7th clock. In relative mode, when the internal time counter reaches to 3, signal state is driven and the counter resets to 0. It then takes 5 more clocks to drive the next signal state value.

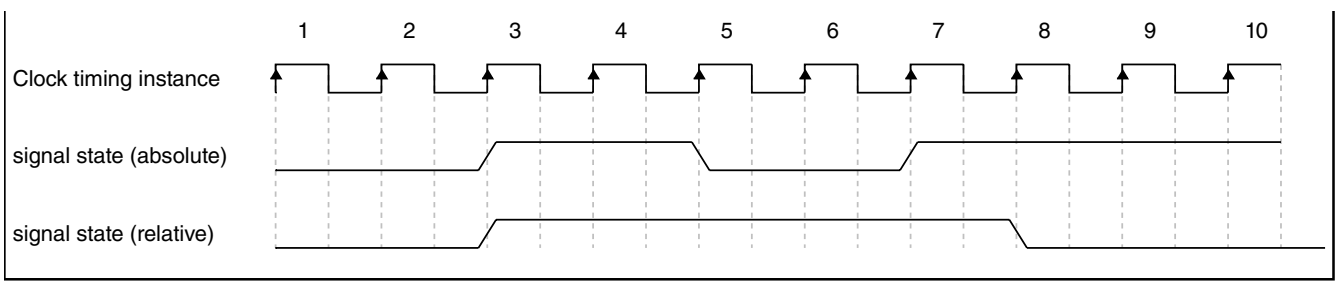


Figure 46-13. Timing entry effect on signal state in relative and absolute mode

46.8.6.1 CPU configuration

CPU can configure one of the two available timing tables so as to enable the CTE to generate various internal and external signals. The CPU always updates the table which is not active. If it tries to update the active timing table, timing table contents will be overwritten and a contention error flag (lut_contn_err) of the Debug Register (CTE_DBG_REG, bit 1) will be asserted. Each timing table will have 32 entries containing timing instances and the signal states corresponding to that particular timing instance for the internal and external signals.

46.8.6.2 eDMA trigger

Timing tables can also be configured using the eDMA trigger functionality. CTE can generate eDMA triggers to the eDMA engine to start a DMA transfer. These trigger events can be configured to occur on toggling of the timing table and the start/end of the timing table. On any of the above events, a DMA trigger will be generated by the CTE. It is the responsibility of the DMA controller to make sure that the correct timing table is configured.

46.8.7 Overflow counter

The overflow counter is a 32bit counter, counting the time of the overflow flags. The counter is reset with radar frame synchronization (RFS) signal at the beginning of each frame start. Two such counters exist, one for ADC overflows and the other for receiver overflows.The implementation of both counters is identical, only the overflow input is connected to different sources as described below.

46.8.7.1 Receiver overflow counter

The receiver overflow input is one selected GPIO input and synchronized to CTE clock domain by two flop stages. It is qualified with LUT0_LSB[rcv_ovf], in order to generate a window in which the overflow is valid. The counter is enabled with this combined signal and counts while it is '1'. The counter value is mapped to configuration register. See the figure below for more details:

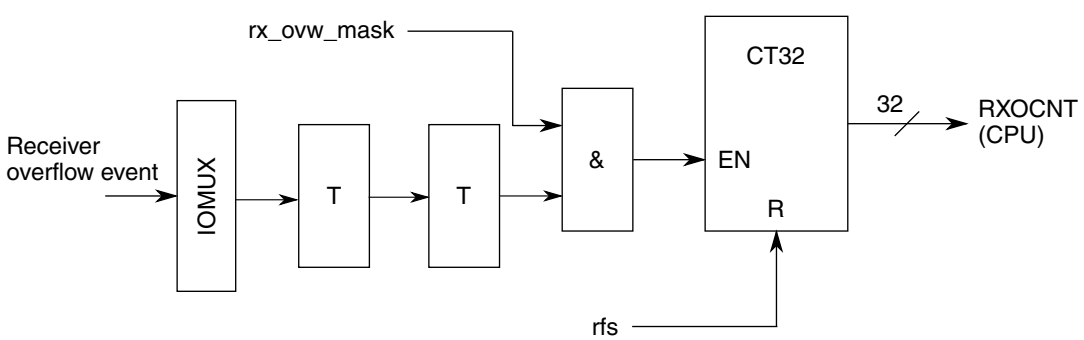


Figure 46-14. Receive overflow counter

46.8.7.2 ADC Overflow Counter

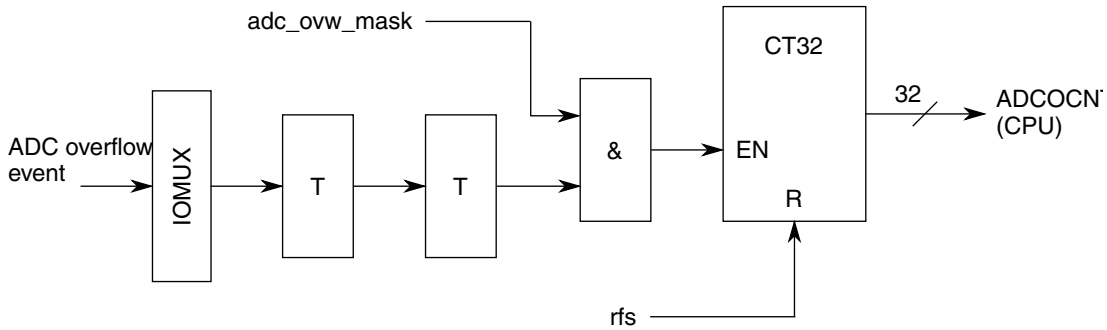


Figure 46-15. ADC Overflow Counter

The ADC overflow counter signal is the result of the OR operation on all the ADC overflow flags. It is qualified with LUT0_LSB[adc_ovf] to generate the counter enable signal.

46.8.8 CrossTrigger Timing

The cross-triggering ensures a timing correlation between the output of the DAC and the capturing of antenna input. Depending on the RADAR system, the generation of timing signals may completely change between RADAR measurement cycles. It may also vary within phases, e.g. to modulate the timing of chirp waveform within a couple of clock cycles.

The following picture shows a typical timing for ADC and DAC triggering for two un-modulated chirp waveforms. See the figure below for more details:

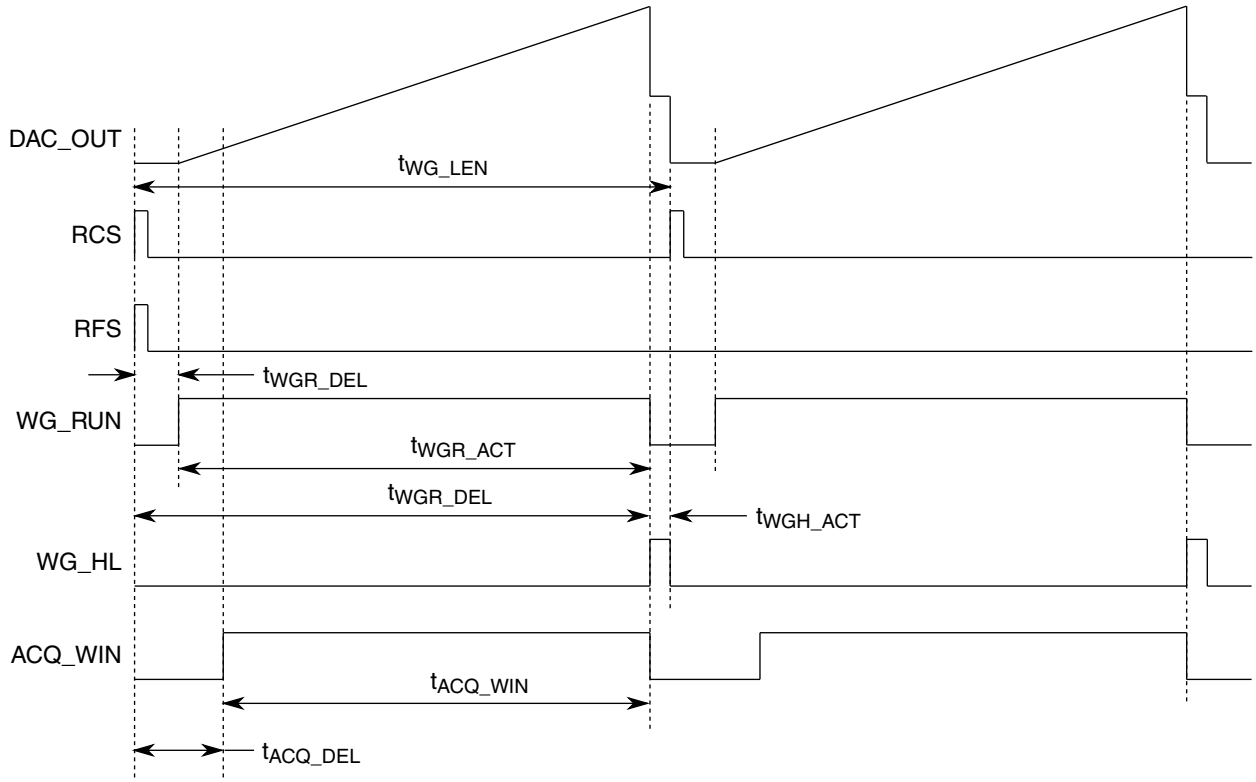


Figure 46-16. Cross Trigger Timing

46.9 Interrupts

CTE can generate interrupts to the CPU on various events e.g. start of each timing table, end of each timing table or by rising edge of selected timing signals. See the interrupt status register CTE_INTSTAT for details. The interrupts can be masked using the corresponding enable bits in interrupt enable register CTE_INTEN. An interrupt is indicated by the event flag if corresponding enable bit is set and cleared by W1C strategy.

46.10 Checksum for LUT0/LUT1

The Checksum register carry the most recent calculated checksum for LUT0 and LUT1. It is calculated on LUT content and signal mode content. The Checksum is updated on each update on any LUT content.

The Checksum registers are in read/write mode. The start Checksum value can be written into the register before making update to the LUT content. The updated content can be read back from the same registers.

Checksum is calculated in two modes:

1. Checksum is calculated for 39:0 bits and carryover at bit 40 is dropped.
2. Checksum is calculated for 39:0 bits and carryover at bit 40 is added back to LSB that is bit0.

Thus, the initial Checksum is loaded into internal registers in the following way:

- Initial_Checksum[40:0] = {1'b0, CTE_CKSM_MSB.CKSM_MSB[7:0], CTE_CKSUM_LSB.CHKS_LSB[31:0]}

The final Checksum is calculated in the following way:

- Final_Checksum[40:0] = Initial_Checksum[40:0] + {1'b0,CTE_SIGTYPE0n[24],CTE_SIGTYPE0n[22:0],CTE_LUT0_LSBn[15:0]};

This Final_Checksum is again updated to Initial_Checksum and on the next write, new Initial_Checksum is taken. On resetting the Checksum, the entire Checksum is reset.

46.11 Timing table configuration example

The modification of active timing table values is possible, but is not recommended. If this is attempted by the SW, the contents in the timing table will be replaced but the behavior of CTE will not be defined (a contention error bit also sets in debug register CTE_DBG_REG.

The recommended way to update the timing table values is to temporarily stop timing generation by disabling the CTE and updating the timing table or update currently inactive timing table by configuring CTE in toggle mode and using timing table end interrupts. With the first method, CPU can prepare all required timing data in main SRAM as a linked list. The timing tables are then loaded by eDMA into the CTE automatically. With the second method, CPU can calculate the next timing data and load it directly into the current unused CTE timing table using the "table_sel" bit in the debug register to know which timing table is currently in use. The minimum time covered by a single timing table is mainly limited by the latency of the eDMA in the first approach and the response time of the CPU in the second approach. See the following figure for more details.

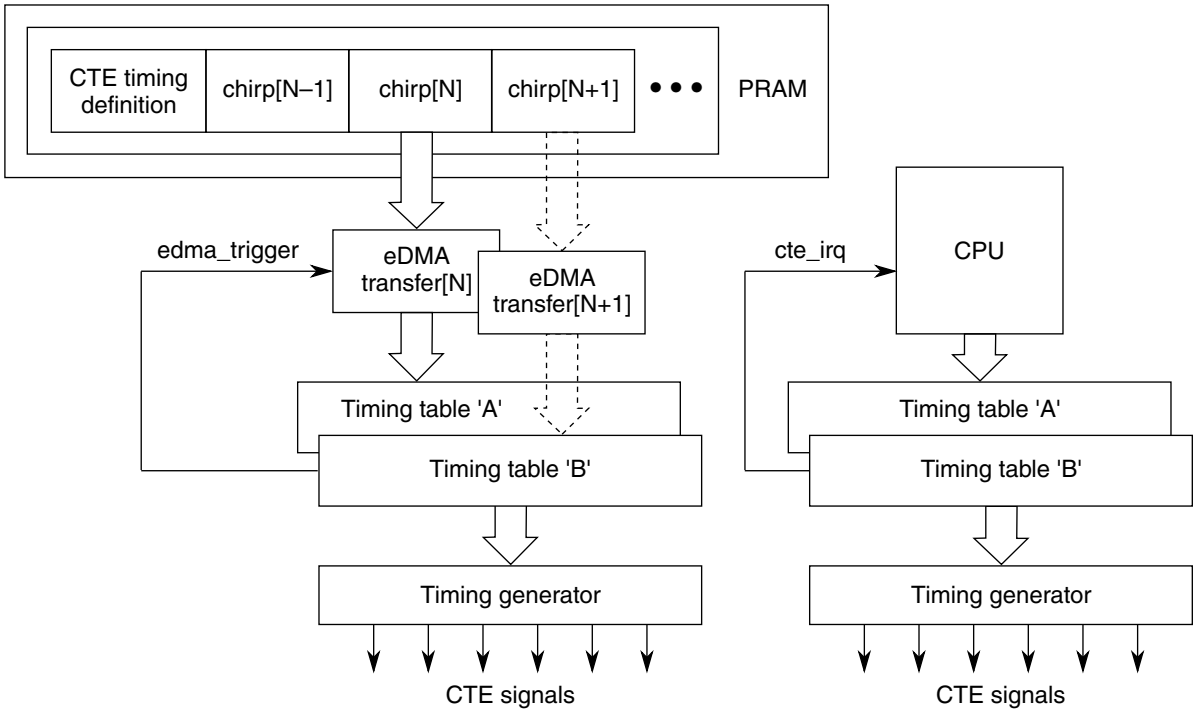


Figure 46-17. eDMA and CPU Programming Model

46.12 Initialization sequence

CTE supports different modes in which the timing table is executed, eDMA trigger is generated etc. Following is the sequence which should be followed by the user in order to configure CTE:

1. Configure time instances and corresponding signal states in the timing table register LUT_LSB_0.
2. Configure remaining signal states in the timing table register LUT_MSB_0.
3. Configure time instances and corresponding signal states in the timing table register LUT_LSB_1.
4. Configure remaining signal states in the timing table register LUT_MSB_1.
5. Configure signal types in signal type registers CTE_SIGTYPE0/1.
6. Configure MA_SL_ST bit in control register CTE_CNTRL.
7. Configure REP_CNT value in control register CTE_CNTRL.
8. Configure OPMOD_SL bits in control register CTE_CNTRL to define continuous run mode or toggle mode.
9. Configure the duration counter for TT0 and TT1 execution in registers CTE_LUT_DUR and CTE_LUT_DUR1 respectively.

NOTE

The user must not program either of the CTE LUT duration registers to "0". See the NOTEs in TT0 Execution Duration Register (CTE_LUT_DUR) and TT1 Execution Duration Register (CTE_LUT_DUR1).

10. Configure timemode bit in CTE_CNTRL1 to define absolute/relative mode of timing table execution.
11. Configure CTECK_DV in CTE_CNTRL1 to define the CTE data path clock.
12. Configure CLKDIV_1-CLKDIV_4 in CTE_CNTRL1 to define clocks for external signals.
13. Configure CLK_SEL0-CLK_SEL9 in clock select register CTE_CLKSEL to select the clock divider for the external signals.
14. Enable CTE by setting CTE_EN bit in CTE_CNTRL1.

Note that steps 1-4 can be done using either eDMA or CPU. Following are the steps for CPU configuration:

- Since there are separate interrupts to the CPU on start and end of both the timing tables, CPU can easily find out which one is being used and which one is free. Using this info, CPU can configure the table which is not being used by the CTE for timing generation without any contention.

In case of eDMA configuration following are the steps:

- configure eDMA_CTL bits in control register CTE_CNTRL. This will configure CTE to generate the trigger either on start or end of the table execution. If start of execution is being used as a trigger, eDMA will have to make sure that it updates the other table in order to avoid contention. In case when end is being used as a trigger, eDMA will have to update the same table.

Configuring Clock type ports

CTE enables user to configure any of the CTEP[7:0], RCS and RFS ports as clock and send periodic signals which are aligned to one of the 4 clock dividers. Suppose user wants to send 10 M signal on CTEP[0] and RCS, 5 MHz signal on RFS and CTEP[1], 2.5 MHz signal on CTEP[4:2] and 1.25 MHz signal on CTEP[7:5]. Following steps should be used:

- Configure CTE_SIGTYPE0 register to define CTEP[7:0] as clock type signals. CTE_SIGTYPE0[CTE_TYPE0] = 0x10, this will configure CTEP[0] as clock, similarly configure rest of the CTEP[7:1] as clock.
- Configure CTE_SIGTYPE1 register to define RCS and RFS as clock type signals. CTE_SIGTYPE1[RCS] = 0x10 and CTE_SIGTYPE1[RFS] = 0x10.

- Configure clock divider 1 to generate 10 MHz clock. CTE_CNTRL1[CLKDIV_1] = 0x011".
- Configure clock divider 2 to generate 5 MHz clock. CTE_CNTRL1[CLKDIV_2] = 0x100".
- Configure clock divider 3 to generate 2.5 MHz clock. CTE_CNTRL1[CLKDIV_3] = 0x101".
- Configure clock divider 4 to generate 1.25 MHz clock. CTE_CNTRL1[CLKDIV_4] = 0x110".
- Configure clock select register so that RCS and CTEP[0] are aligned to clock divider 1 and are sending 10 MHz signal. CTE_CLKSEL[CLK_SEL8] = 0x00 and CTE_CLKSEL[CLK_SEL0] = 0x00.
- Configure clock select register so that RFS and CTEP[1] are aligned to clock divider 2 and are sending 5 MHz signal. CTE_CLKSEL[CLK_SEL9] = 0x01 and CTE_CLKSEL[CLK_SEL1] = 0x01.
- Configure clock select register so that CTEP[4:2] are aligned to clock divider 3 and are sending 2.5 MHz signal. CTE_CLKSEL[CLK_SEL2] = 0x10, CTE_CLKSEL[CLK_SEL3] = 0x10 and CTE_CLKSEL[CLK_SEL4] = 0x10.
- Configure clock select register so that CTEP[7:5] are aligned to clock divider 4 and are sending 1.25 MHz signal. CTE_CLKSEL[CLK_SEL5] = 0x11, CTE_CLKSEL[CLK_SEL6] = 0x11 and CTE_CLKSEL[CLK_SEL7] = 0x11.

- Enable CTE by setting CTE_EN bit in CTE_CNTRL1.

Chapter 47
Waveform Generation Module (WGM)

47.1 Introduction

The WGM generates the digital output using one of the four configurable look-up tables (LUT), to be converted into the waveforms by the DAC module. The waveforms can be programmed by means of four configurable LUTs. The timing and control signals for generating these LUT outputs to the DAC is provided by the Cross Triggering Engine (CTE) module.

The following table lists various acronyms and their description that are used in WGM module.

Table 47-1. Acronyms

Acronym	Description
WGM	Waveform Generator Module
DAC	Digital-to-Analog Converter
LUT	Look-up Table
CTE	Cross Triggering Engine
FSM	Finite State Machine
GPIO	General Purpose I/O
PWM	Pulse-width modulation
SW	Software
LFSR	Linear Feedback Shift Registers
FADR	Fixed Address
CVAL	Counter Value
MISR	Multi Input Shift Register

47.2 Block Diagram

The block diagram of the WGM module is shown below:

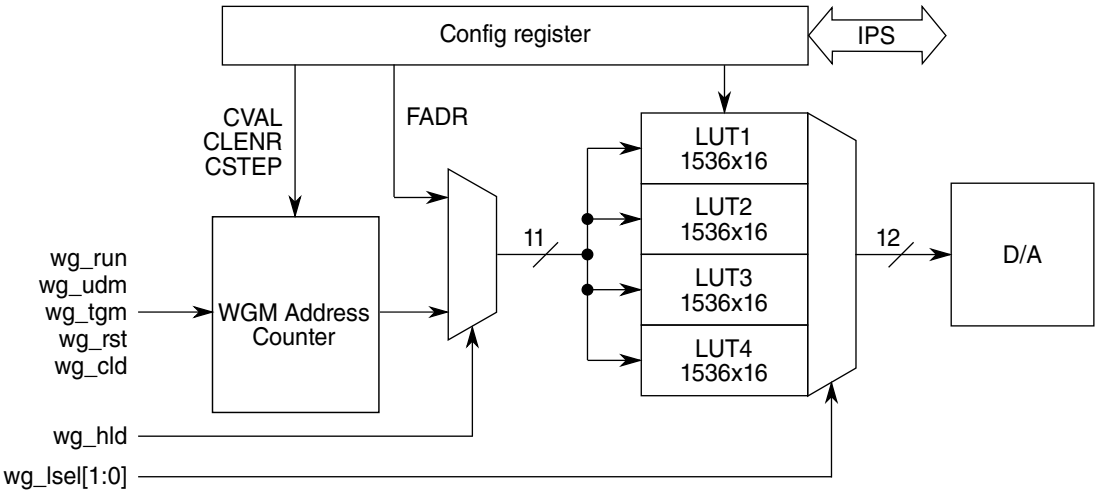


Figure 47-1. WGM Block Diagram

47.3 Features

The WGM module has the following key features

- Programmable LUT based chirp generation.
- Up to four 16x1536 configurable LUTs provided.
- 5 Programmable PWM waveform generation modules.
- WGM LUT does not have ECC protection.
- Automatic update of waveform data by eDMA.
- Stand alone mode supported.

47.4 Interrupts

The WGM module can generate interrupts to the CPU on the following events:

- At the start of LUT execution i.e. when WGM starts reading the LUT and WGM address counter is 0.
- At the end of the LUT execution when WGM is reading the LUT and WGM address counter either equals or exceeds last entry in the LUT or (WGM Control Register 1 (WGM_CTRL_1) [VALID_CNT]).

These interrupts are masked and will show on the interrupt line only when the corresponding enable bit is set in the INT_EN [31:30] and the event occurs. Clearing of the interrupts follows w1c strategy.

47.5 Modes of operation

There are different scenarios under which different types of data from a selected LUT has to be sent to the DAC from the WGM. Based on the wg_lsel bits coming from the CTE, active LUT for sending the data to the DAC is selected. The data to be sent to the DAC is stored in LUTs. An 11-bit WGM address counter is used to read the data from the LUT. For generating different types of data to the DAC, WGM address counter needs to be controlled.

Following are the modes in which WGM logic operates to generate different values of WGM address counter and hence different types of data to the DAC:

- Hold Mode
- Off Mode
- Stop Mode
- Run Mode
- Stand alone mode

Different modes are described below:

47.5.1 Hold Mode

In this mode, a constant value is sent to the DAC module. This constant value is stored in specific locations of all the LUTs. WGM address counter keeps on running the way it is configured but the address for reading the LUT data is loaded with the Fixed address from the CPU (FADR) which corresponds to the above specific location. In this way a particular LUT location is selected to send a fixed value to the DAC. There is a Multiplexer which ensures that the FADR is used as the LUT read address instead of the WGM address counter in hold mode.

47.5.2 Off Mode

In this mode, output to DAC is disabled i.e. all zeroes are transmitted to the DAC from WGM. WGM address counter is in reset. WGM enters in this mode after reset and remains in this mode until WGM is enabled by software.

47.5.3 Stop Mode

In this mode, a fixed value is transmitted to the DAC from WGM. This is achieved by freezing the WGM address counter once the Stop mode is entered causing the same value to be transmitted until the WGM is in this mode.

47.5.4 Run Mode

In this mode the LUT address counter runs in control of the CTE (Cross Triggering Engine). Following signals shown in the table, for example, wg_udm, wg_tgm, wg_rst, wg_cld decide in which mode the LUT address counter should be running. Depending upon the signals coming from CTE module, WGM address counter value is generated.

Table 47-2. WGM Address Counter Mode Table

Signal Name	Description
wg_udm	When this signal is '0, WGM address counter decrements by step size "cstep" on every clock and then rounds off to 0 after reaching the end address1535. When "wg_udm" is '1', WGM address counter resets to 1535 and then starts incrementing by "cstep" on every clock until it reaches 0 where it rounds off to 1535 again. Please note that in either mode, round off will take the counter to 0 e.g. suppose wg_udm is 1 and cstep is 30. Counter reaches 1530 then the next value will be 0 and then 30 and so on. Similar will be the case when wg_udm is 0. Please see WGM Address Counter Example where some examples are shown depicting various WGM address counter run scenarios.
wg_tgm	When this signal is '1', the counter increments by "cstep" on one clock and then decrements by the same on the next clock.
wg_rst	When this signal is asserted, WGM address counter will be reset to 0 on the next WGM clock cycle.
wg_cld	When this is set, WGM address counter will be loaded with "cval" on the next WGM clock.
cval	This is the value with which the counter will be loaded on the next WGM clock when "wg_cld" signal is asserted.
cstep	This is the step size with which the counter will increment or decrement depending upon the "wg_udm".
clenr	This the factor which limits the data rate from WGM to DAC. By default, this is 0 which means that the data rate will be 80 Mega samples per seconds. Value of clenr is taken in power of 2 which means that if it is 1, the data rate will become 40 Mega samples per seconds, if it is 2, data rate will become 20 Mega samples per seconds, if it is 3, data rate will become 10 Mega samples per seconds and so on. The Supported Data Rates are 10 Msps and Lower. User should configure it in a way that the rate should remain 10 Msps or Lower for a DAC.

47.5.5 Standalone Mode

Normally the WGM modes of operation is controlled by CTE but there is a provision to control the WGM operation by CPU in stand alone mode (by setting CTRL_1[11]). In this case, the WGM operation is started and stopped by CPU. Once started, WGM address counter increments until it reaches the end value and then rounds off to value 0. It always runs in incremental mode and increments always by 1.

NOTE

Please note that only LUT1 is available in this mode.

47.6 WGM Address Counter Example

The time at which the control signals become effective on the address counter and the DAC data is different depending upon states of the signals. Following table lists the number of clock cycles it takes for a particular control signal to affect the address counter. From the time address counter is changed, DAC data takes 3 WGM clock cycles to show up. LUT number and location depends upon the "wg_lsel[1:0]" and address counter value respectively. The following table shows the number of clock cycles required by the control signals from CTE, to affect the WGM address counter.

Table 47-3. Timing table relating to address counter and control signals

Control Signal	Required clock cycles to change the address counter
wg_run	1
wg_tgm	1
wg_cld	1
wg_udm	1
wg_rst	1
wg_hld	0

It can be seen from the table above that except "wg_hld" which affects the counter immediately/asynchronously, all the other control signal affect the counter synchronously i.e. take effect on the next clock cycle.

Following example diagrams show the counter output dependent upon the signals coming from the CTE. Please note that the frequency of clock shown below in all the diagrams will be according to the CLENR value programmed in CTRL_1 [26:29].

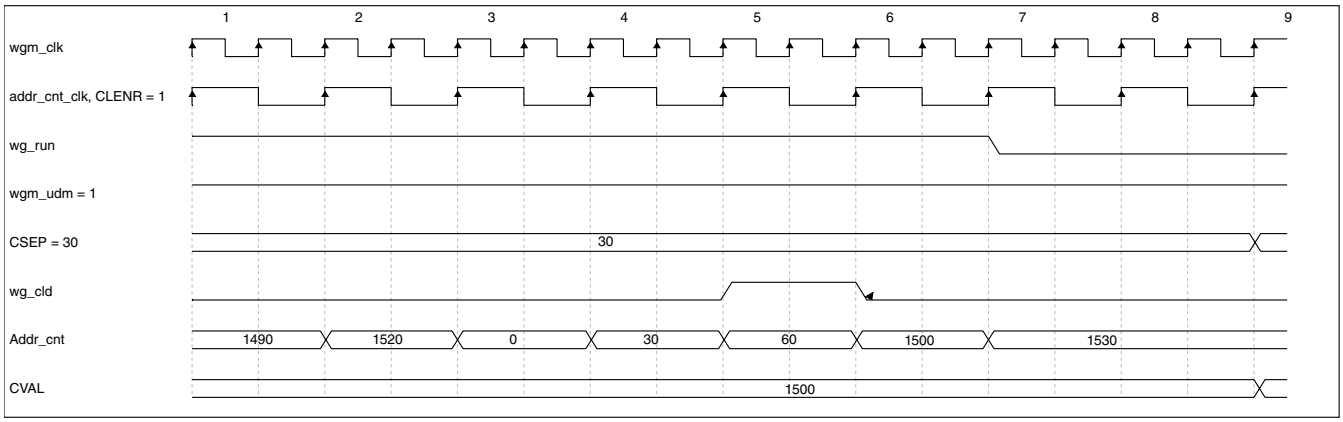


Figure 47-2. Incremental mode with CSTEP as 30 and wg_cld

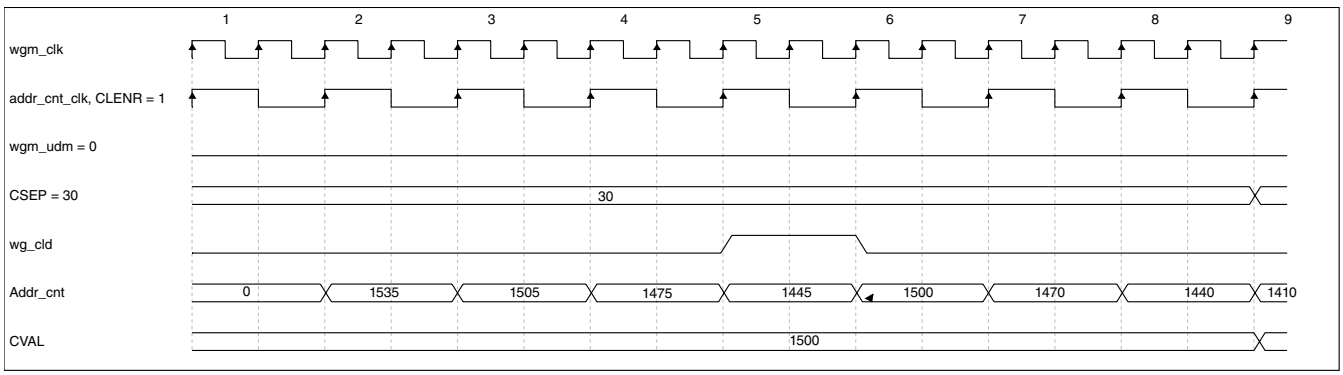


Figure 47-3. Decremental mode with CSTEP as 30 and wg_cld

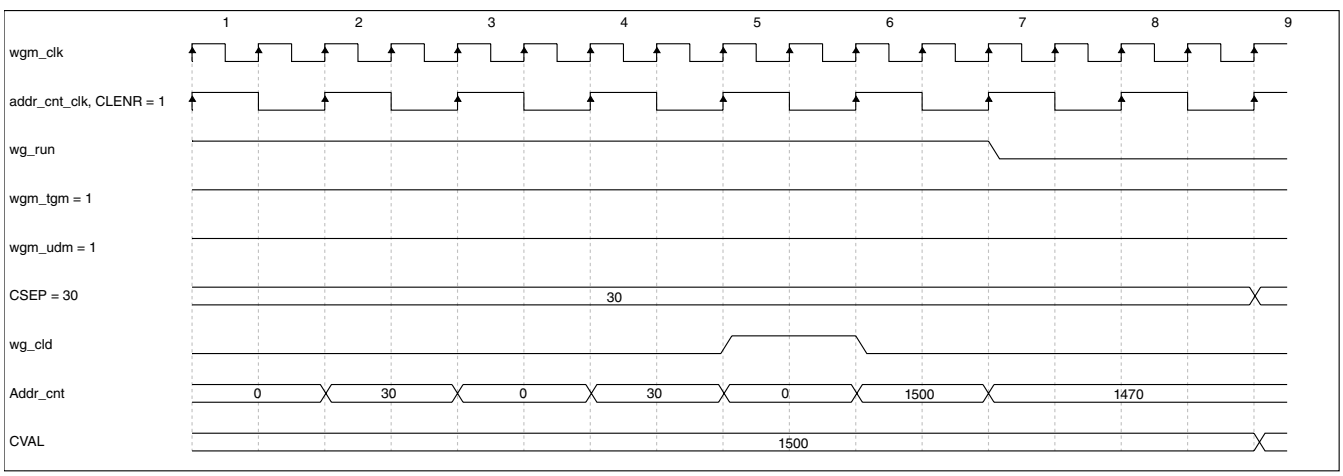


Figure 47-4. Incremental mode with CSTEP as 30 and wg_tgm and wg_cld

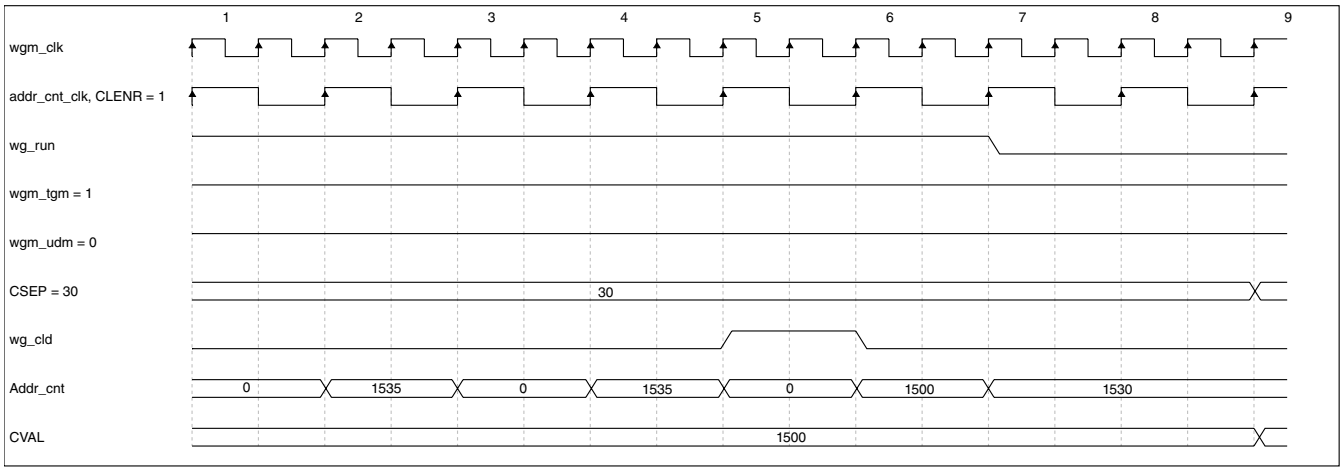


Figure 47-5. Decremental mode with CSTEP as 30 and wg_tgm and wg_cld

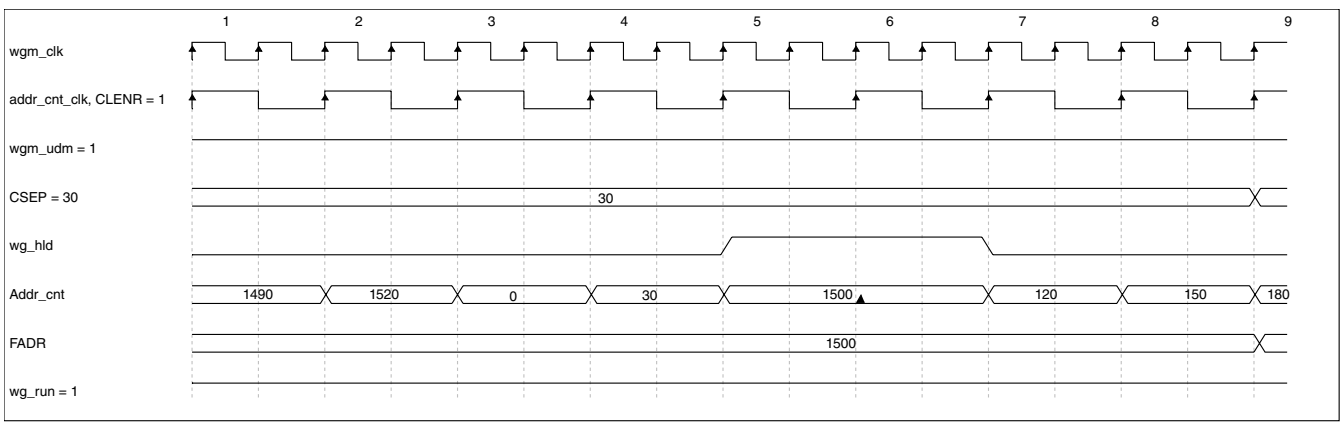


Figure 47-6. Incremental mode with CSTEP as 30 and wg_hld and wg_run = 1

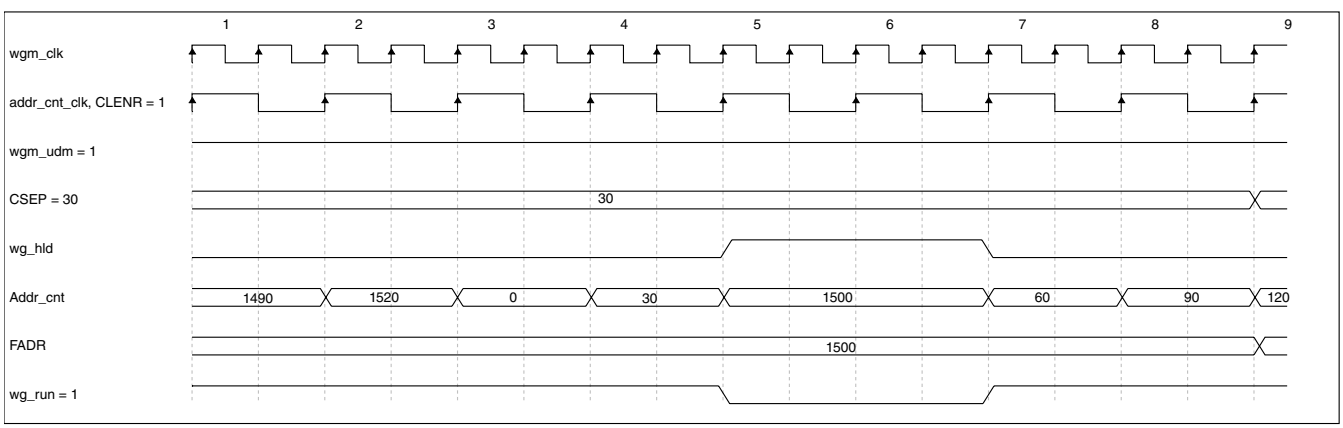


Figure 47-7. Decremental mode with CSTEP as 30 and wg_hld and wg_run = 0

Figure 47-8 below shows that the WGM module starts sending data to the DAC module after three clock cycles as soon as it is enabled by software. Which LUT will be used, is decided by "wg_lsel[1:0]" value and LUT address location will be decided, based on the address counter value which is 0 by default.

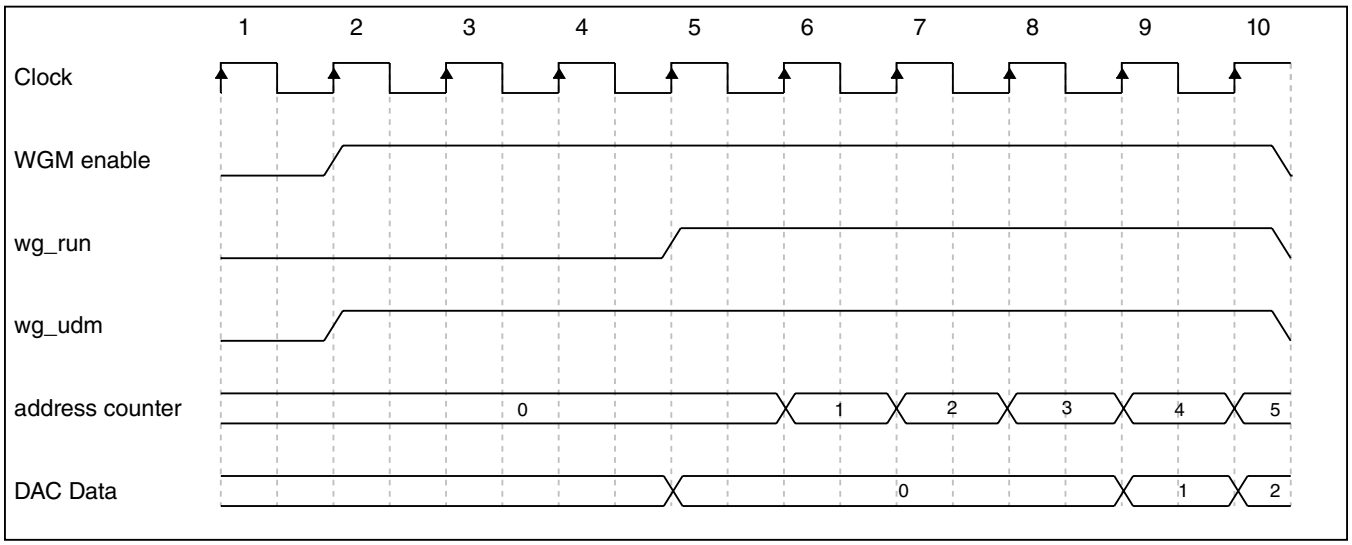


Figure 47-8. DAC data behavior with WGM enable, wg_run

Figure 47-9 below shows the address counter behaviour and the data which will be sent to the DAC when the LSEL along with other control signals are changing. Please note that after WGM is enabled, it will take 3 WGM clocks for WGM module to send data to the DAC if dac_ade is 0. If dac_ade is 1, it will take 4 clock cycles.

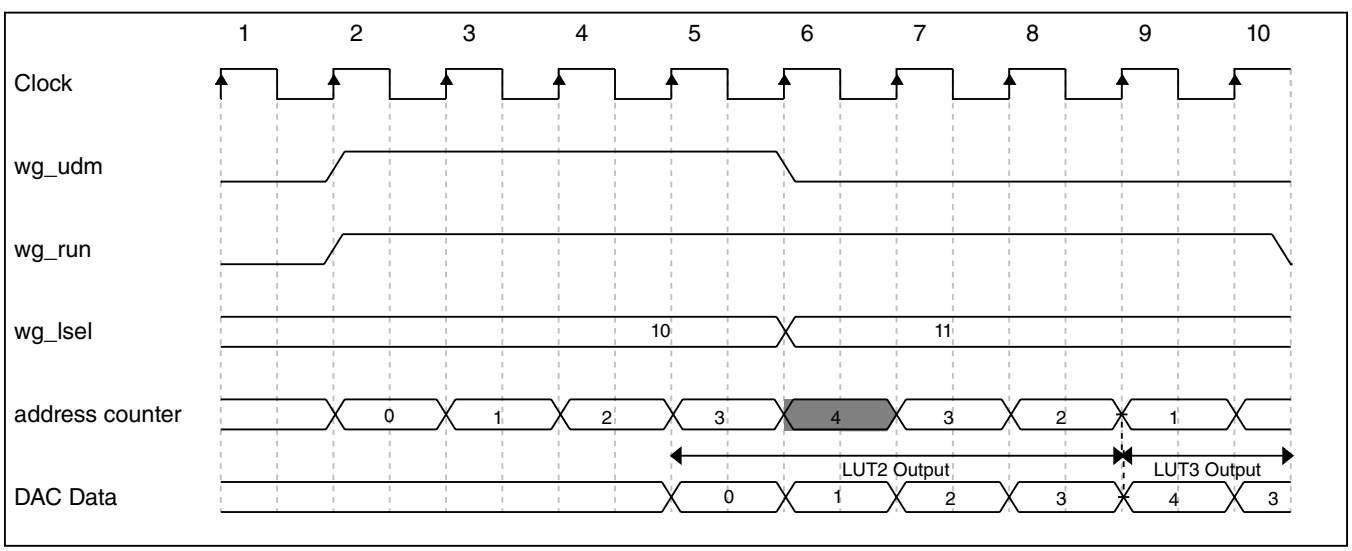


Figure 47-9. Address counter and DAC data relation in case of LSEL change

Figure 47-10 below shows the address counter behavior and the data which will be sent to the DAC when the wg_rst is asserted. Since address counter is already at 0, it is appearing that wg_rst is taking effect for more than 4 clocks. Shaded area of the DAC data output shows the values which was sent corresponding to reset.

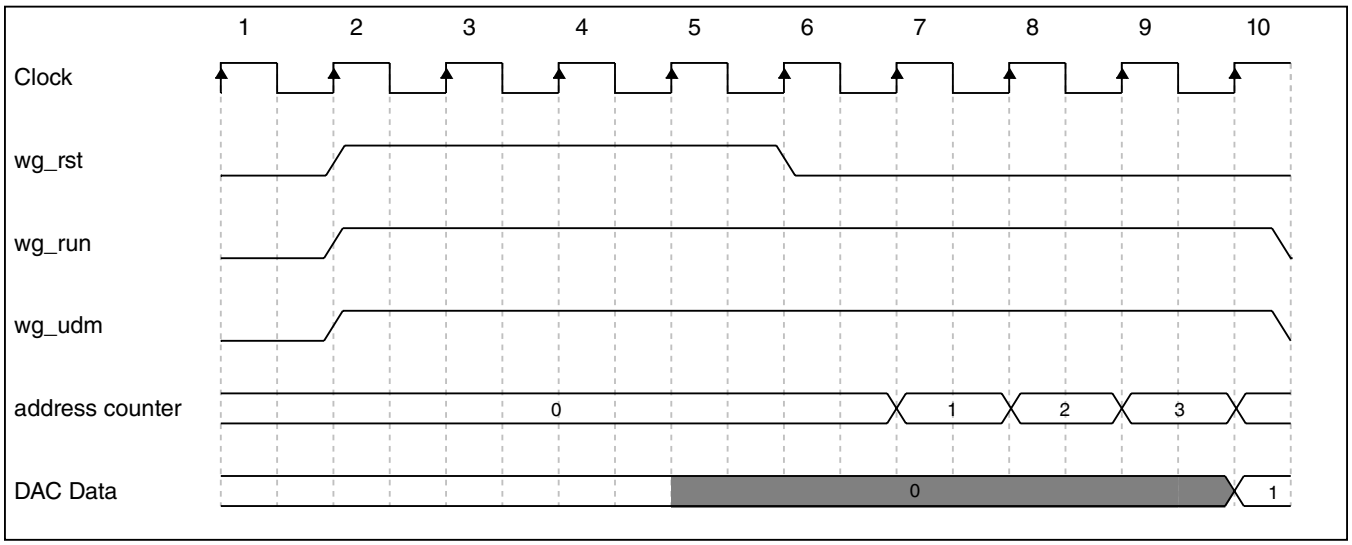


Figure 47-10. Address counter and DAC data relation with wg_rst

Figure 47-11 shows when wg_rst is asserted when address counter has a non-zero value.

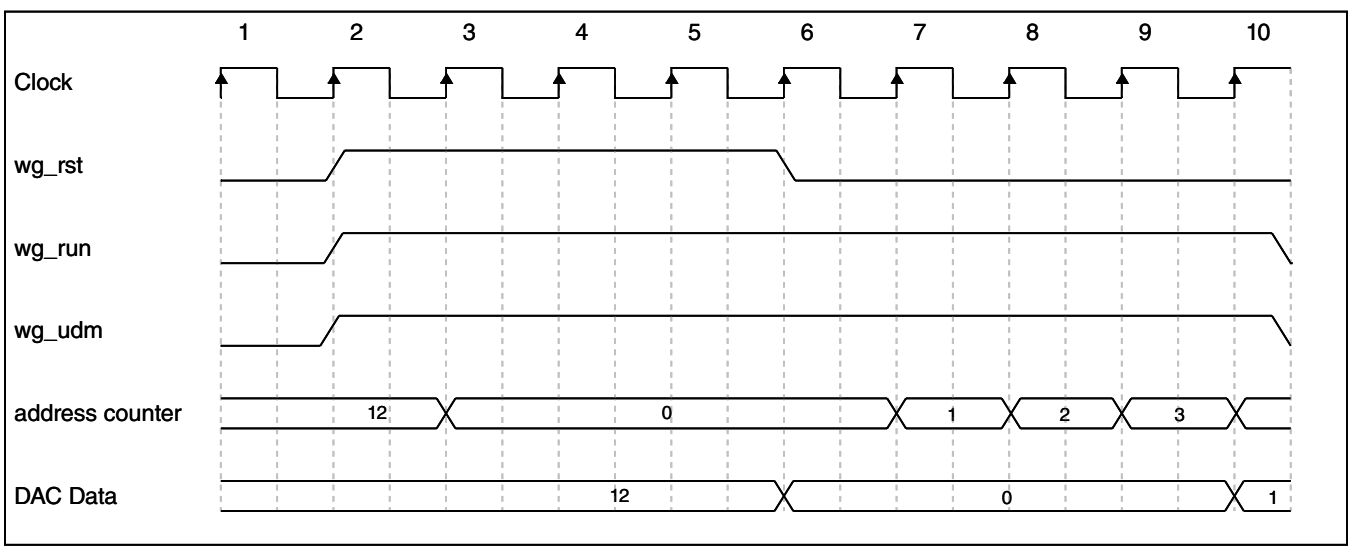


Figure 47-11. Address counter and DAC data relation with wg_rst and non-zero count value

Figure 47-12 shows the address counter and associated DAC data output behavior when wg_cld is asserted.

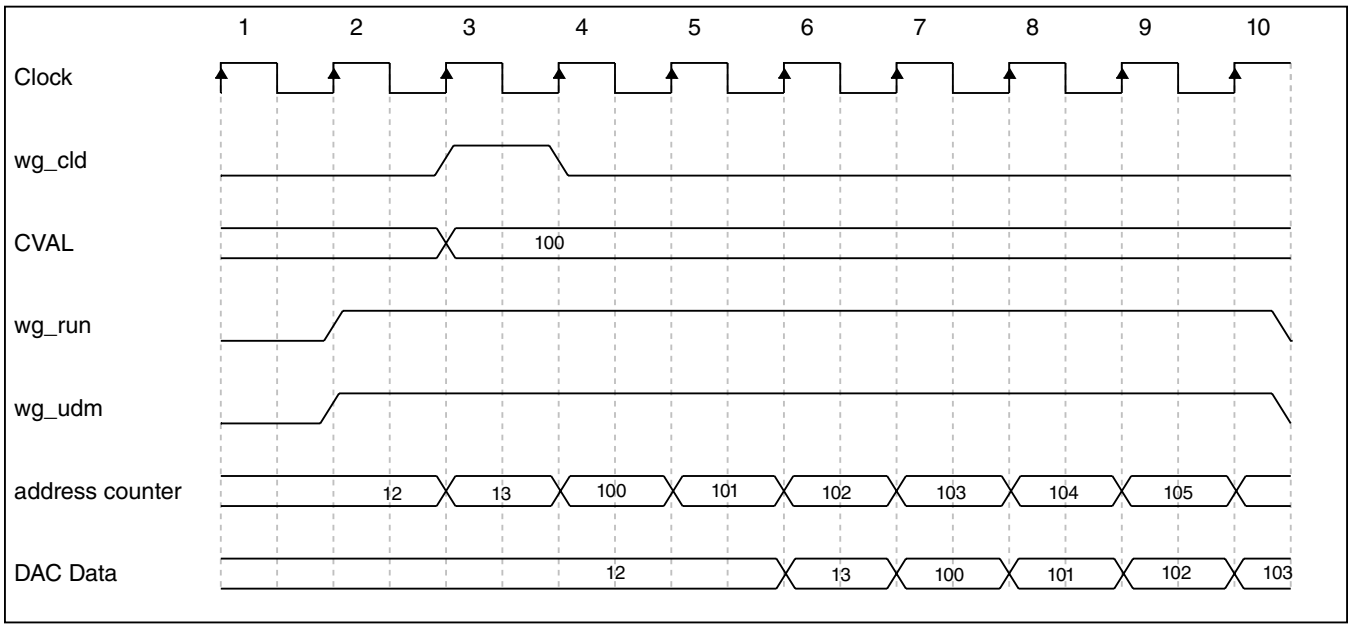


Figure 47-12. Address counter and DAC data relation with wg_cld

Figure 47-13 shows the case where in "wg_run" signal from CTE is high just for 1 WGM clock and CLENR is 2 meaning that the address counter is running at divide by 4 clock. wg_run fails to increment the address counter but wg_run1 increments it. It is the responsibility of the SW to ensure that the timing signal is high for at least one divided clock period as per CLENR value.

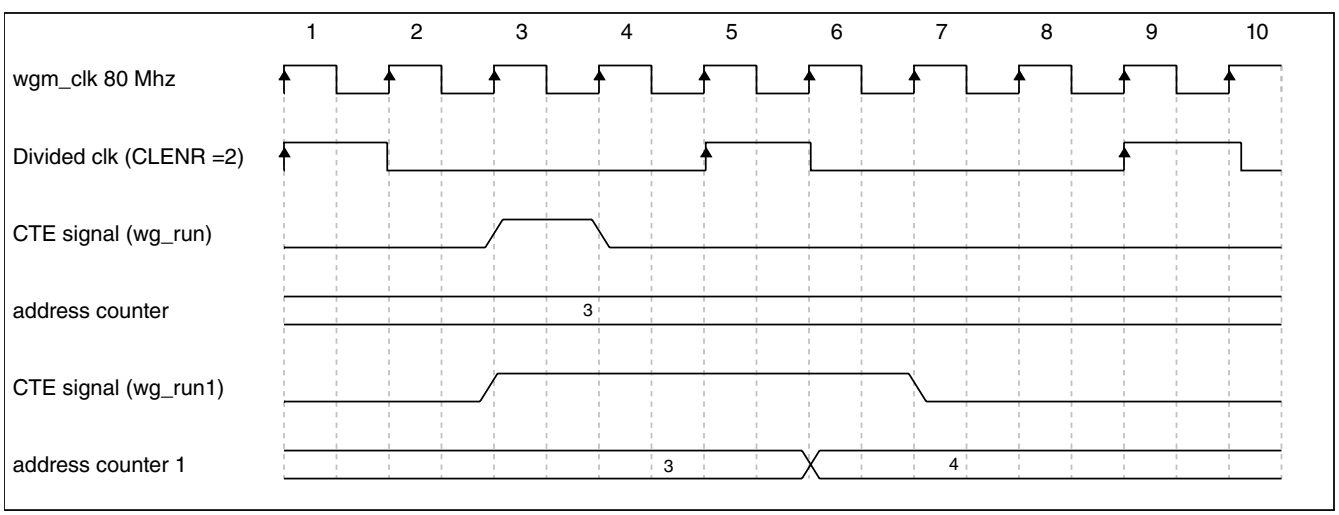


Figure 47-13. Address counter behaviour with timing signal when CLENR is high

Figure 47-14 below show the effect of WGM_CTRL_1[WG_HLD_S]. When this bit is '1', wg_hld signal's behavior becomes synchronous, that is, it's effect is delayed by one clock. This means that when wg_lsel[1:0] is changing from 0 to 1, data corresponding to FADR starts showing up after one clock and for that clock data from LUT0 shows up at

the DAC data output. When WGM_CTRL_1[WG_HLD_S] is '0', the effect of wg_hld is asynchronous and hence the data corresponding to FADR starts showing up at the data output on the same clock.

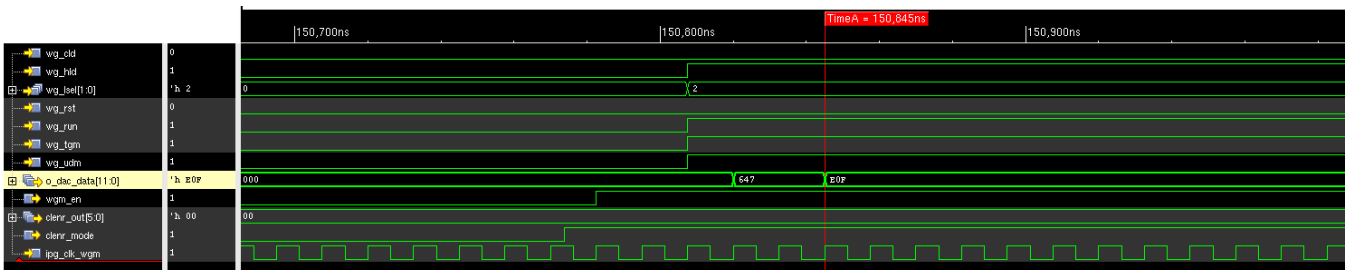


Figure 47-14. wg_hld synchronous behavior

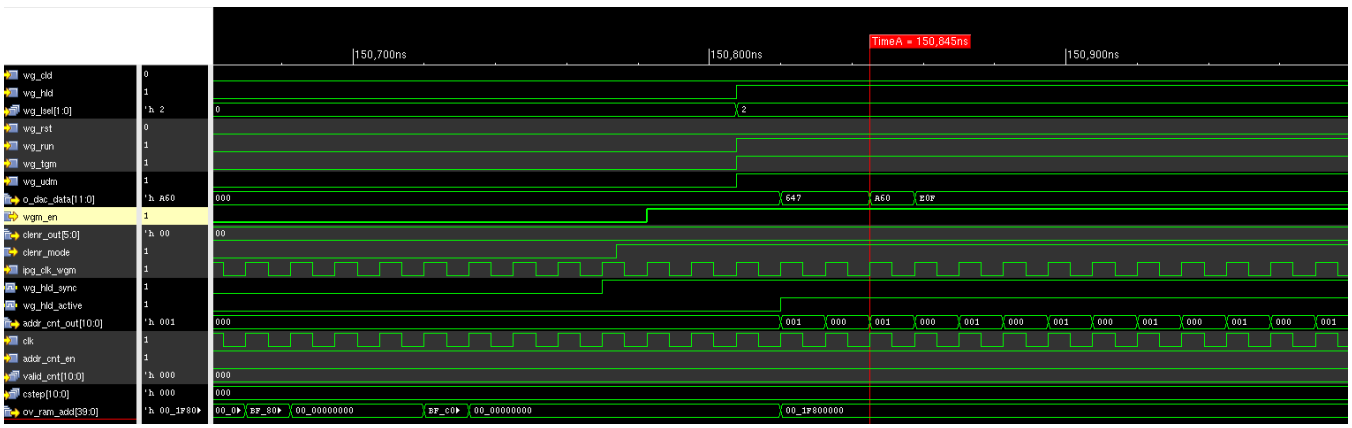


Figure 47-15. wg_hld asynchronous behavior

47.7 Memory Map and Register Description

WGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	WGM Control Register (WGM_CTRL)	32	R/W	0000_0000h	47.7.1/2168
4	WGM Control Register 1 (WGM_CTRL_1)	32	R/W	0000_0000h	47.7.2/2171
8	CVAL Register (WGM_CVAL)	32	R/W	0000_0000h	47.7.3/2173
C	Fixed Address Register (WGM_FADR)	32	R/W	0000_0000h	47.7.4/2174
10	LUT Address Register (WGM_LUT_ADDR)	32	R/W	0000_0000h	47.7.5/2174
14	LUT Data Register (WGM_LUT_DATA)	32	R/W	0000_0000h	47.7.6/2175
18	WGM LFSR128 Register (WGM_LFSR_128)	32	R/W	0000_0000h	47.7.7/2176
1C	LFSR Select Register (WGM_LFSR_SEL)	32	R/W	0000_0000h	47.7.8/2177
20	WGM Interrupt Enable Register (WGM_INT_EN)	32	R/W	0000_0000h	47.7.9/2178

Table continues on the next page...

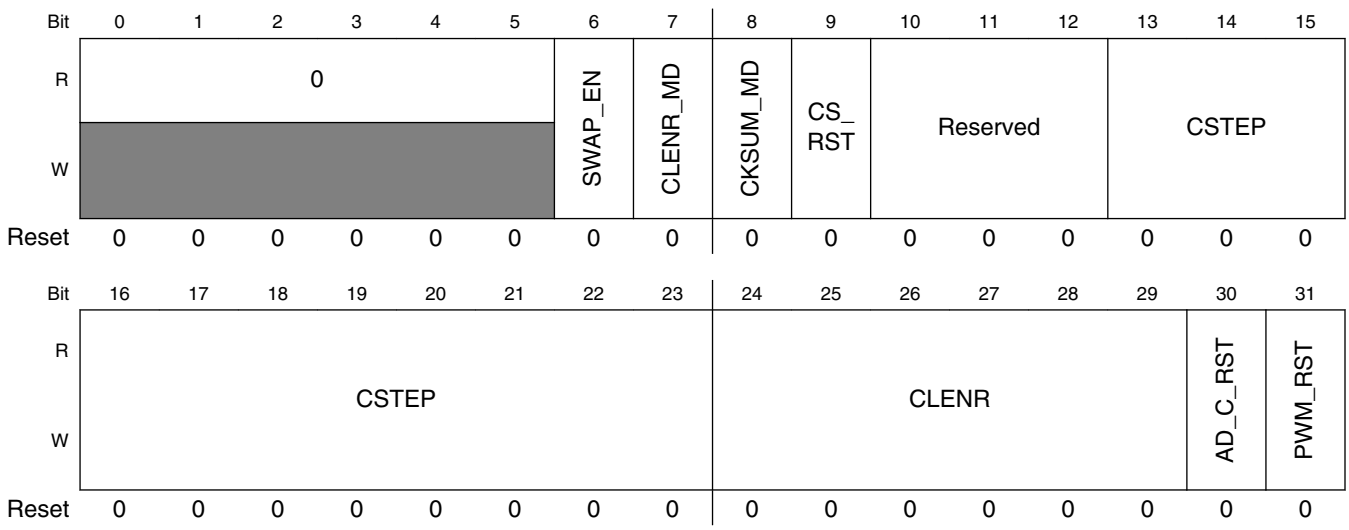
WGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	WGM Interrupt Status Register (WGM_INT_STAT)	32	w1c	0000_0000h	47.7.10/ 2180
28	LUT Checksum Register (WGM_LUT_CKSM)	32	R/W	0000_0000h	47.7.11/ 2181
2C	Debug Register (WGM_DEBUG)	32	w1c	0000_0000h	47.7.12/ 2182

47.7.1 WGM Control Register (WGM_CTRL)

This is the first control register of WGM module.

Address: 0h base + 0h offset = 0h



WGM_CTRL field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SWAP_EN	SWAP Enable This bit is used to swap the endianness of the data which is programmed into the WGM LUTs. 0 No swapping is done while programming the data onto the LUTs. 1 Swapping is performed on 16-bit boundary while programming the data on LUTs. For example, if swapping is enabled and the content 0xabcdbcda is written on WGM_LUT_Data, then the content which will be written onto the LUT will be 0xbcdaabcd.
7 CLENR_MD	CLENR Mode

Table continues on the next page...

WGM_CTRL field descriptions (continued)

Field	Description
	Thit bit decides the mode in which the value of WGM_CTRL [CLENR] is taken to limit the rate at which WGM transmits data to the DAC. The maximum data rate supported is 10 Msps. User should configure in a way that the maximum data rate does not exceed 10Msps. 0 CLENR value is taken in power of 2 i.e. the way it is described in CLENR bit field description. Only 4 lower bits are used in this case. 1 CLENR value is taken in decimal e.g. when WGM_CTRL[CLENR] is "101000", the data rate to DAC is limited to 2 Mega samples per second.
8 CKSUM_MD	Checksum Calculation Mode This bit decides whether checksum calculator block works in checksum mode or MISR (multi input shift register) mode. 0 Checksum mode. Carry on MSB is dropped. For example: current value = 1000000000000000 next write value = 1000000000000000 checksum result = 0000000000000000 (Carry on MSB is dropped). 1 MISR mode, carry from MSB is added to the LSB. For example: current value = 1000000000000000 next write value = 1000000000000000 checksum result = 0000000000000001 (carry on MSB is added to LSB).
9 CS_RST	Checksum Reset This bit when asserted causes the Checksum calculation block to reset synchronously on the next IPG clock. 0 De-asserted. 1 Checksum block will be synchronously reset on the next IPG clock cycle.
10–12 Reserved	This field is reserved.
13–23 CSTEP	Counter Step CSTEP defines the step by which WGM address counter should increment/decrement depending upon the wg_run and wg_udm signals from CTE. It can have values from 0 to 2047. **NOTE:** Please note that if the value of CSTEP is programmed to be more than the LUT DEPTH and the current value of wgm address counter is non-zero, it will roll over to 0 and stay there. If the current value is 0, the counter anyways will stay at 0. 0000 Address counter increments/decrements by 1. .. 1111 Address counter increments/decrements by 16 and so on.
24–29 CLENR	Counter LENR CLENR defines the rate at which the WGM transmits data to the DAC. Depending upon the CLENR mode bit (WGM_CTRL[CLENR_MD], either only 4 lower bits are taken in the power of 2 as per the table below, or all 6 bits are taken in decimal. For example if user programmes WGM_CTRL[CLENR] = "101000" and WGM_CTRL[CLENR_MD] is 0, then the value taken will be 2**8 which is 256 and DAC data will be limited to 80/256 = 0.3125 MSPS. IF WGM_CTRL[CLENR_MD] is 1, then value will be taken as 40 and DAC data will be limited to 80/40 = 2 MSPS. **NOTE:** Please note that it is the responsibility of the SW to ensure that the CLENR is correctly programmed in order to match the data rate of WGM to the internal DAC in order to avoid data loss. The maximum data rate supported is 10 Msps. User should configure in a way that the maximum data rate does not exceed 10 Msps. 0000 Reserved

Table continues on the next page...

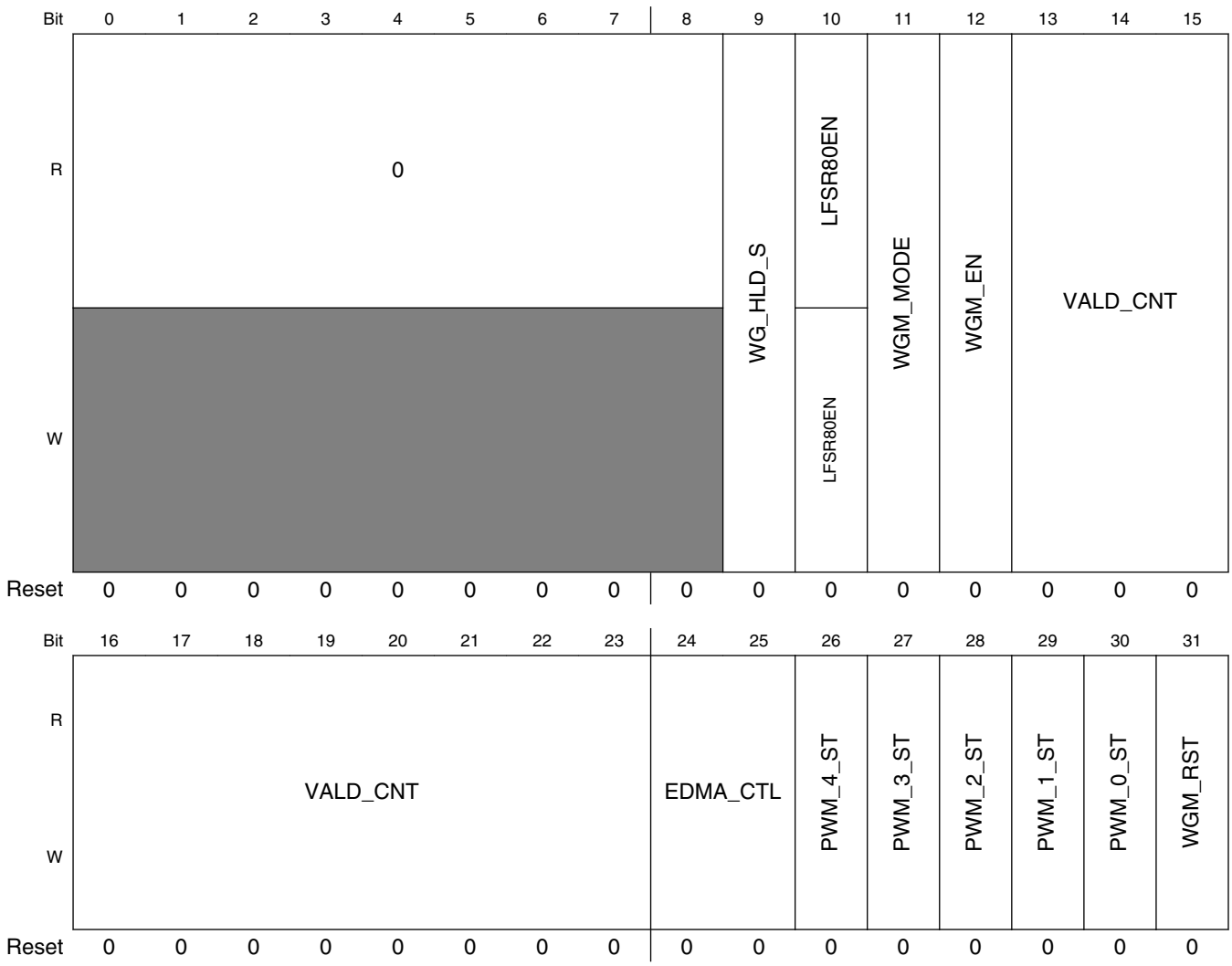
WGM_CTRL field descriptions (continued)

Field	Description
	0001 Reserved 0010 Reserved 0011 Data rate is 10 M Samples per second. 0100 Data rate is 5 M Samples per second. 0101 Data rate is 2.5 M Samples per second and so on.
30 AD_C_RST	Address Counter Reset This bit is used to reset the WGM address counter separately. This bit does not have any effect on other modules. This can be used for debugging purpose. 0 De-asserted. 1 WGM address counter is reset synchronously on the next WGM clock cycle.
31 PWM_RST	PWM Reset This bit is used to reset the 5 LFSR modules. This bit does not have any effect on other modules. This can be used for debugging purpose. **NOTE:** Please note that this bit does not reset the PWM module start bits (PWM_4_ST, PWM_3_ST, PWM_2_ST, PWM_1_ST and PWM_0_ST) in the WGM_CTRL1 register. Hence user should first reset the individual PWM start bit before asserting the PWM_RST and then set it individually after de-asserting the PWM_RST. 0 De-asserted. 1 All the 5 PWM generator modules are reset synchronously on the next LFSR clock cycle.

47.7.2 WGM Control Register 1 (WGM_CTRL_1)

This is the second control register of WGM module.

Address: 0h base + 4h offset = 4h



WGM_CTRL_1 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 WG_HLD_S	WG_HLD Synchronous/Asynchronous This bit decides if the effect of wg_hld signal coming from CTE will be synchornous or asynchronous. **NOTE:** Please see the example figure at section "WGM Address Counter Example" for details.

Table continues on the next page...

WGM_CTRL_1 field descriptions (continued)

Field	Description
	0 Effect of wg_hld will be asynchronous i.e. the address to the LUT will be replaced by FADR on the same WGM clock. 1 Effect of wg_hld will be synchronous i.e. the address to the LUT will be replaced by FADR on the next WGM clock.
10 LFSR80EN	LFSR 80 Mhz Clock Enable This bit is used to select the clock at which LFSRs operate. 0 LFSRs operate at 40 Mhz. 1 LFSRs operate at 80 Mhz.
11 WGM_MODE	WGM mode This bit is used to select the mode in which the WGM address counter operates. 0 WGM operation is dependent upon the control signals coming from CTE. 1 WGM operates in stand alone mode. In this mode, the address counter starts from 0 and increments by 1 until the it reaches 1535 and then rolls back to 0. It always runs in incremental mode. Only one LUT (LUT1) is available in this case.
12 WGM_EN	WGM Enable This bit is used to enable the transmission of data to the DAC. When this is 0, zeroes are transmitted to the DAC. WGM address counter will retain its value and the operation mode will go back to OFF mode. 0 WGM operation will remain in OFF mode meaning that zeroes will be transmitted to the DAC. WGM address counter will retain the current value. 1 WGM operation is started. In normal mode the operation will be dependent upon the control signals coming from the CTE.
13–23 VALD_CNT	Valid Entry Counter This field describes how many entries in the LUT are valid. The value of WGM address counter will be rolled over to 0 once it becomes greater than or equal to this value. If this value is 0, it is considered as 1535 meaning all the entries in the table are valid. **NOTE:** Please note that the value of "VALD_CNT" should always be <=LUT DEPTH.
24–25 EDMA_CTL	eDMA trigger control These 2 bits decide which event is used to send the eDMA trigger for LUT configuration. 00/11 eDMA trigger generation disabled. 01 eDMA trigger is generated when the active LUT which is being used to send data to the DAC is started i.e. the value of WGM address counter is 0. 10 eDMA trigger is generated when the active LUT which is being used to send data to the DAC reaches the end i.e. the value of WGM address counter is either >= 1535 or it is >= WGM_CTRL_1[VALD_CNT] at 0x0004).
26 PWM_4_ST	5th PWM module start This bit is used for starting/stopping the PWM generator. For configuring the seed, it should first be stopped. 0 Third 360-bit PWM generation module is stopped. 1 Third 360-bit PWM generation module is running.

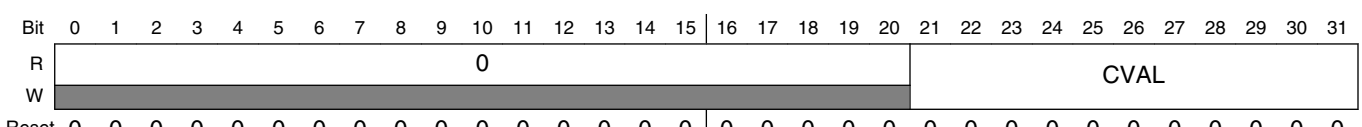
Table continues on the next page...

WGM_CTRL_1 field descriptions (continued)

Field	Description
27 PWM_3_ST	4th PWM module start This bit is used for starting/stopping the PWM generator. For configuring the seed, it should first be stopped. 0 Second 360-bit PWM generation module is stopped. 1 Second 360-bit PWM generation module is running.
28 PWM_2_ST	3rd PWM module start This bit is used for starting/stopping the PWM generator. For configuring the seed, it should first be stopped. 0 First 360-bit PWM generation module is stopped. 1 First 360-bit PWM generation module is running.
29 PWM_1_ST	2nd PWM module start This bit is used for starting/stopping the second PWM generator. For configuring the seed, it should first be stopped. 0 Second 128-bit PWM generation module is stopped. 1 Second 128-bit PWM generation module is running.
30 PWM_0_ST	1st PWM module start This bit is used for starting/stopping the first 128 bit PWM generator. For configuring the seed, it should first be stopped. 0 First 128-bit PWM generation module is stopped. 1 First 128-bit PWM generation module is running.
31 WGM_RST	WGM Reset This bit is used to reset the WGM module synchronously. 0 Reset is de-asserted. 1 All the WGM modules will be reset synchronously on their next respective clocks.

47.7.3 CVAL Register (WGM_CVAL)

Address: 0h base + 8h offset = 8h



WGM_CVAL field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

WGM_CVAL field descriptions (continued)

Field	Description
21–31 CVAL	Counter Value This is the value to which counter is initialized on the next WGM clock cycle when "wg_cld" signal is asserted by the CTE. **NOTE:** Please note that the value of CVAL should always be less than or equal to the LUT DEPTH in order to send a valid data to the DAC.

47.7.4 Fixed Address Register (WGM_FADR)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																					FADR_VAL										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WGM_FADR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 FADR_VAL	Fixed Address Normally wgm address counter value is used as the address for reading the data from active LUT (depending upon the "wg_lsel" value). But when "wg_hld" signal from CTE is 1, FADR_VAL is used as the address counter on the same clock. **NOTE:** Please note that the value of FADR should always be less than or equal to the LUT DEPTH in order to send a valid data to the DAC.

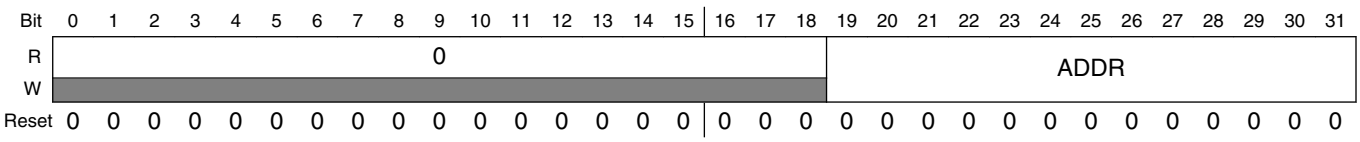
47.7.5 LUT Address Register (WGM_LUT_ADDR)

This register contains the address along with the LUT select bits corresponding to the 2 data values in the LUT_DATA register.

NOTE

Do not access this register when the WGM_CLK is disabled. As this clock is derived from the Sigma Delta PLL (SDPLL) the SDPLL must be on in the current mode before accessing this register.

Address: 0h base + 10h offset = 10h



WGM_LUT_ADDR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–31 ADDR	LUT address value This address is used for updating the LUT by the CPU. LUT depth is 768 which means 10 bits are required for addressing the LUT. ADDR [22:31] will be used for LUT addressing and ADDR [20:21] will be used as LUT select and ADDR [19] is unused. So the location of the LUT select bit is not fixed but is dependent upon the LUT DEPTH. Next 2 bits after the addressing bits are always taken as the LUT select bits. For example, suppose LUT DEPTH is 64, then ADDR [26:31] will be used for addressing, ADDR [24:25] will be used as LUT select, and ADDR [19:23] will be unused. Address is loaded by the CPU only once and then, auto-increments by 1 after every write to the LUT_DATA register. **NOTE:** The implemented LUT dimension is 1536 deep and 16-bit wide (1536x16). However, since the CPU data bus width is 32, in order to minimize the time taken by the CPU to configure the LUT, the dimension is taken as 768 deep and 32-bit wide.

47.7.6 LUT Data Register (WGM_LUT_DATA)

This register is accessed by the CPU to configure the LUTs. The value written in this register gets loaded into the LUT at location mentioned in the LUT address register. This register contains data for 2 locations of LUT selected by the LUT select bits and address counter. For example, if CPU writes 0x000 in the address register, internal address counter will be loaded with 0 and LUT0 will be selected. DAC_VALUE [16:27] will be written to LUT location 0 and DAC_VALUE [0:11] will be written to LUT location 1. Then on every write on the data register, internal address counter increments by 1 and the data is similarly written on the next LUT location.

NOTE

In case of 16-bit writes to the data register, skip writing will take place. Please refer Lookup Table Configuration.

NOTE

Do not access this register when the WGM_CLK is disabled. As this clock is derived from the Sigma Delta PLL (SDPLL) the SDPLL must be on in the current mode before accessing this register.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DAC_VAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WGM_LUT_DATA field descriptions

Field	Description
0–31 DAC_VAL	DAC value for LUT

47.7.7 WGM LFSR128 Register (WGM_LFSR_128)

This register is used to program the seed for the PWM modules. The contents of this register update the LFSR seed depending upon which LFSR is selected and which DWORD, i.e., 32-bit chunk of the selected LFSR is selected. Please see LFSR select register at address 0x001C for details. Normally for configuring 128 LFSR seed it requires four 32-bit write access + some wait states (introduced because IPG clock and WGM clocks are asynchronous). In 8-bit mode, it will take 16 write access + wait states. For configuring 360-bit LFSR in 8/16-bit mode will take even longer. Hence this register is designed to be accessed in 32-bit mode only.

NOTE

CPU should always configure the seed register in 32-bit write access mode. If CPU tries to access this register in 8/16-bit mode, entire 32-bit seed will be modified by the write data bus content leading to incorrect results. Hence it is recommended to programme the seed in 32-bit write mode only.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	LFSR_VAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

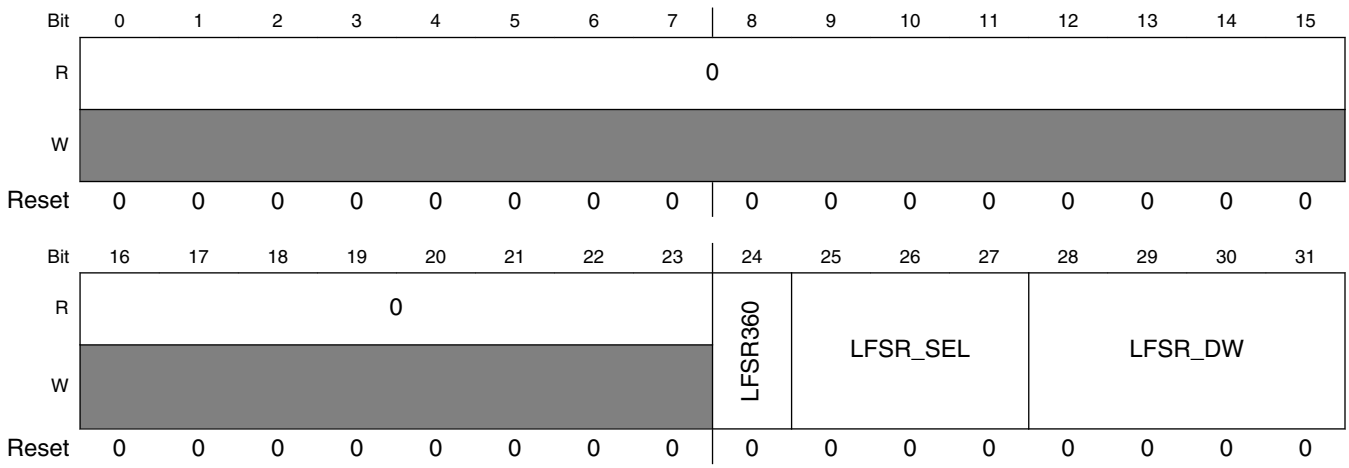
WGM_LFSR_128 field descriptions

Field	Description
0–31 LFSR_VAL	1st LFSR128 Value This value/pattern is loaded by the CPU as seed to the selcted PWM module. Corresponding "pwm_start" bit in WGM_CTRL1 should be 0 for programming the seed value. The PWM generator runs forever once it is started and is stopped only for loading the new seed value.

47.7.8 LFSR Select Register (WGM_LFSR_SEL)

There are total 5 PWM/LFSRs modules. 3rd, 4th and 5th are 360-bit and 1st and 2nd are 128-bit. This register selects which LFSR module out of the 5 is used for updating the seed.

Address: 0h base + 1Ch offset = 1Ch



WGM_LFSR_SEL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LFSR360	LFSR360 MUX Select This bit is used to select between the positive output of the second 360-bit LFSR and negative output of the third 360-bit LFSR. 0 Positive (p) output of third 360-bit LFSR module is connected to forth LFSR360 output of the WGM module. Please see PWM Generator section under Functional Description for details. 1 Negative (n) output of second 360-bit LFSR module is connected to forth LFSR360 output of the WGM module. Please see PWM Generator section under Functional Description for details.
25–27 LFSR_SEL	LFSR Select These bits will select which LFSR will be updated out of the 5 available LFSRs. 000 No LFSR is selected for updating the seed. 001 First 128-bit LFSR is selected for updating the seed. 010 Second 128-bit LFSR is selected for updating the seed. 011 First 360-bit LFSR is selected for updating the seed. 100 Second 360-bit LFSR is selected for updating the seed. 101 Third 360-bit LFSR is selected for updating the seed. 110-111 No LFSR is selected for updating the seed.
28–31 LFSR_DW	LFSR Double Word Select These 4 bits select which double word of the selected LFSR gets configured.

Table continues on the next page...

WGM_LFSR_SEL field descriptions (continued)

Field	Description	
	0000	Content of the LFSR_128_REG0 will configure the first double word [31:0] of the LFSR selected by the "LFSR_SEL" bits.
	0001	Content of the LFSR_128_REG0 will configure the first double word [63:32] of the LFSR selected by the "LFSR_SEL" bits.
	0010	Content of the LFSR_128_REG0 will configure the first double word [95:64] of the LFSR selected by the "LFSR_SEL" bits.
	0011	Content of the LFSR_128_REG0 will configure the first double word [127:96] of the LFSR selected by the "LFSR_SEL" bits.
	0100	Content of the LFSR_128_REG0 will configure the first double word [159:128] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	0101	Content of the LFSR_128_REG0 will configure the first double word [191:160] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	0110	Content of the LFSR_128_REG0 will configure the first double word [223:192] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	0111	Content of the LFSR_128_REG0 will configure the first double word [255:224] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	1000	Content of the LFSR_128_REG0 will configure the first double word [287:256] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360 bit LFSRs).
	1001	Content of the LFSR_128_REG0 will configure the first double word [319:288] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	1010	Content of the LFSR_128_REG0 will configure the first double word [351:320] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	1011	Content of the LFSR_128_REG0 will configure the first double word [359:352] of the LFSR selected by the "LFSR_SEL" bits (only valid for 360-bit LFSRs).
	1100-1111	Unused. Writing these bits will have no effect on the LFSR configuration.

47.7.9 WGM Interrupt Enable Register (WGM_INT_EN)

This is the interrupt enable register for WGM.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														L_END_EN	L_STR_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

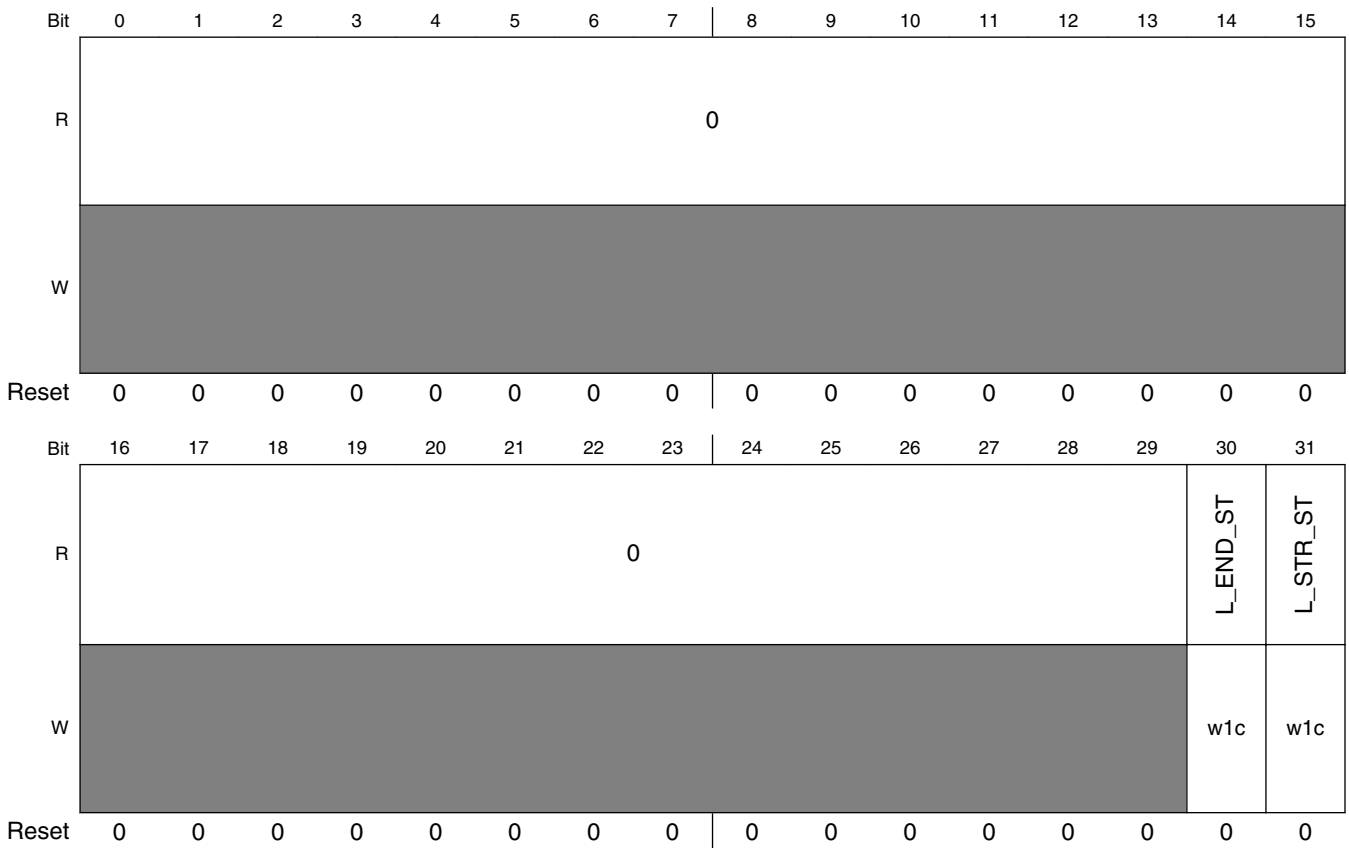
WGM_INT_EN field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 L_END_EN	LUT end enable When this bit is set along with the corresponding event on WGM_INT_STAT an interrupt is generated to the CPU. 0 Disabled. 1 LUT end interrupt will be generated on the corresponding event.
31 L_STR_EN	LUT start enable When this bit is set along with the corresponding event on WGM_INT_STAT register, an interrupt is generated to the CPU. 0 Disabled. 1 LUT start interrupt will be generated on the corresponding event.

47.7.10 WGM Interrupt Status Register (WGM_INT_STAT)

This is the interrupt status register for WGM. All the bits in this register are sticky bits which set on the corresponding events defined in the Interrupt enable register and are cleared by writing a '1'. If the corresponding enable bit in the INT_EN register is set, an interrupt is generated to the CPU.

Address: 0h base + 24h offset = 24h



WGM_INT_STAT field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 L_END_ST	LUT end This bit is set when the WGM is reading the LUT's and the address counter value becomes >= LUT DEPTH or valid entry counter (WGM_CTRL_1 [VALD_CNT]). It is cleared by writing a '1' on it. 0 LUT execution has not finished. 1 LUT execution has finished.
31 L_STR_ST	LUT start

Table continues on the next page...

WGM_INT_STAT field descriptions (continued)

Field	Description
	This bit is set when WGM starts reading the LUT and the address counter value is 0. This can be cleared by writing a 1 on it. 0 LUT execution has not been started. 1 LUT execution has started.

47.7.11 LUT Checksum Register (WGM_LUT_CKSM)

This register contains the 32-bit checksum value for a particular location of the LUT. Its purpose is for the CPU to ensure if the contents were correctly programmed into the LUTs. After the LUT is programmed, CPU reads the checksum value and compares against its own calculated value to ensure correct LUT programming.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CKSUM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

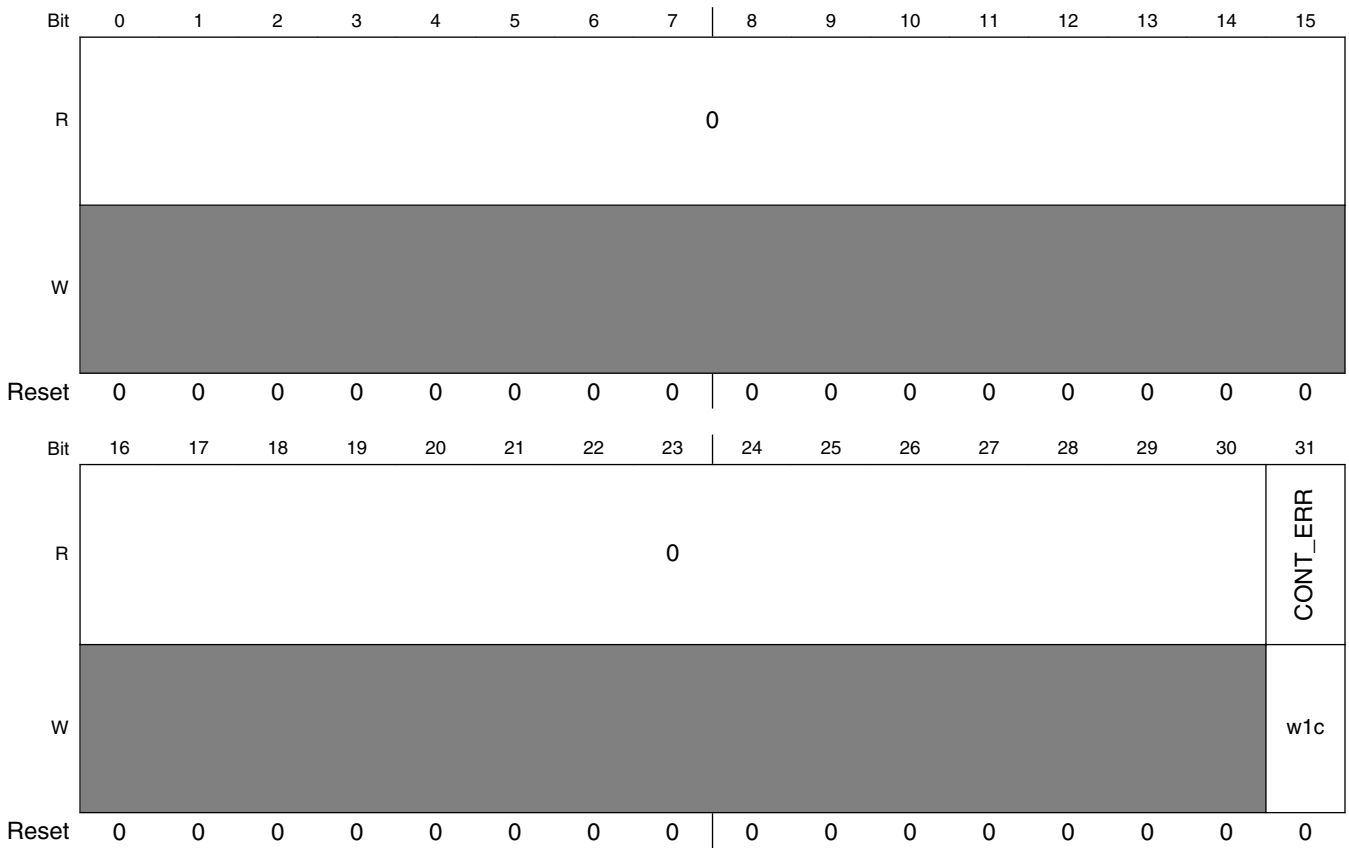
WGM_LUT_CKSM field descriptions

Field	Description
0–31 CKSUM	Checksum These 32-bit contain the checksum corresponding to the bits programmed in the LUT. CPU uses this checksum value to ensure that the LUT was properly configured. Please see Checksum Calculation block for more details.

47.7.12 Debug Register (WGM_DEBUG)

This register is used for the debug purpose.

Address: 0h base + 2Ch offset = 2Ch



WGM_DEBUG field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CONT_ERR	LUT contention error This bit is set when a LUT, which is currently being read by the WGM for sending the data to the DAC, is attempted to be programmed/written by CPU, eDMA,or by any other means. It is cleared by writing a 1 on it. 0 No LUT contention has occurred. 1 LUT contention has occurred.

47.8 Functional Description

Main functionality of WGM module is to run under the supervision of the CTE module and generate the DAC output from one of the 4 programmed LUTs. Main blocks of this module are address counter block and the PWM generator block.

47.8.1 Waveform LUT

The WGM module contains 4 LUTs and multiplexer for LUT output. The LUTs contain the binary numbers for the DAC. Each LUT has 1536 entries of unsigned numbers. The bit-width of the numbers in LUT is 12 bits. Only write operation can be supported from IPS bus, in order to update the LUT values.

47.8.2 Waveform LUT configuration

It is possible to configure the contents of the LUTs using either the CPU or the eDMA.

47.8.2.1 CPU Configuration

It is possible to configure the contents of any of the four LUTs using the CPU. The CPU should ensure that the LUT is inactive before modifying its contents. If it tries to update a LUT while it is active, new data will be loaded but the behavior of WGM will not be guaranteed so this should be avoided by the CPU. In order to let CPU know if any such event occured, a contention error flag (WGM_DEBUG [31] of Debug register) will be asserted. To optimize the performance of the WGM it is possible to update one of the LUTs while any of the other three are active.

47.8.3 eDMA Trigger

There are situations in which the LUT in WGM module need to be directly updated by the eDMA. This is accomplished using a DMA channel using eDMA interface. The WGM module request the eDMA transfer by sending trigger events to the DMA engine. These triggers are configurable (using CTRL_1 [24:25]) and can be sent out on either on the start or end of the LUT (address counter 0 or 1535) or a fixed configurable value of address counter (CTRL_1 [VALID_CNT]). When WGM_CTRL_1[24:25] is not "00/11",

a DMA request signal will be asserted which will cause DMA to configure the LUT. Once DMA is done, it will generate acknowledgement signal which will clear the DMA request signal.

47.8.4 Resets

In order to give control to the SW, there is a provision to assert a software reset bit (WGM_CTRL_1 [WGM_RST]) which will reset the entire WGM module. The effect of this bit is same as the hardware reset. Sometimes, it is required to reset some modules separately. There is a provision to reset the PWM/LFSR generator and Address counter block separately by SW. There are three separate reset signals which are provided for PWM generator WGM_CTRL [PWM_RST], address counter WGM_CTRL[AD_C_RST] and checksum calculation block WGM_CTRL[CS_RST] for this purpose. This may also be useful in debugging.

47.8.5 PWM Generator

There are 5 different Linear Feedback Shift Registers (LFSR) available in WGM module. Two of them (LFSR1-0) are 128 bits and rest are 360 bits (LFSR4-2). There is a provision to stop/start the shifting operation using WGM_CTRL_1 [26:30]. Shifting always happens from MSB to LSB and hence LSB gets shifted out first. Stop is used to load the LFSR and then it is started.

Total 6 outputs are coming out of the LFSR modules. LFSR0 and LFSR1(128 bits) have single bit output. LFSR2 has 2 different output. Positive output of LFSR3 and negative output of LFSR4 are multiplexed (360 bits) to have 2 different outputs. Second output is the inverted of the first output.

Table 47-4. LFSR module connection to the port

LFSR Output port	LFSR Module
lfsr[0]	LFSR0 output
lfsr[1]	LFSR1 output
lfsr[2]	LFSR2 normal output
lfsr[3]	LFSR2 inverted output
lfsr[4]	LFSR3 positive output
lfsr[5]	Multiplexed between the inverted output of LFSR3 and normal output of LFSR4. Inverted output of LFSR4 is unconnected. Please see the figure below for more details.

All these outputs can be enabled/disabled using the GPIO (general purpose input/output) pin multiplexing.

The clock source of this module is 40 MHz, which is achieved by dividing the waveform generator clock (80 MHz) by 2. Please see figure below for more details.

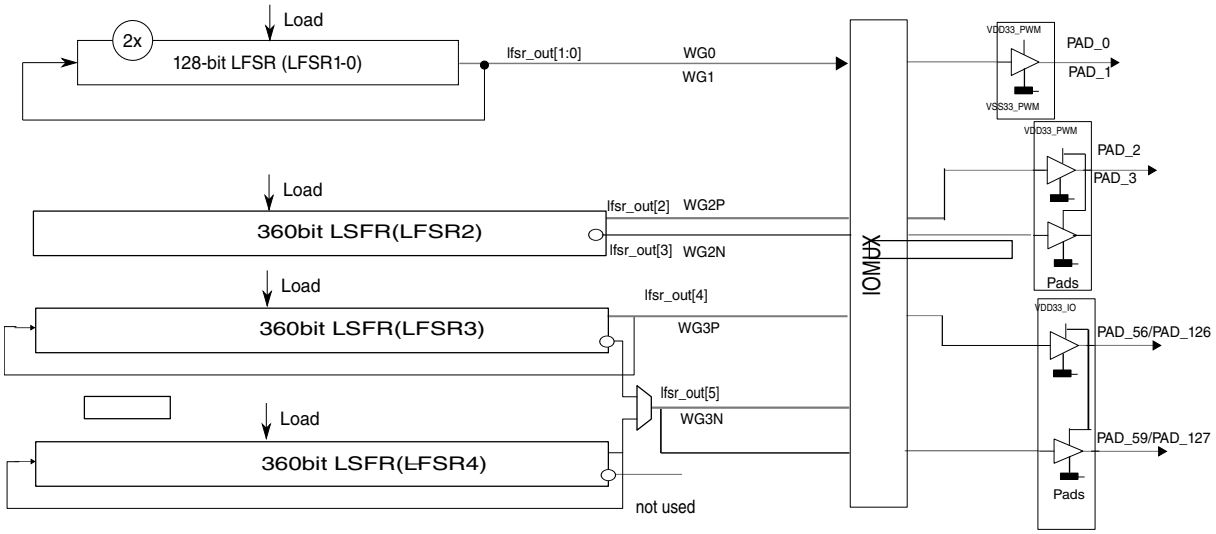


Figure 47-16. PWM Generator

47.8.6 Checksum Calculation

The checksum calculator block is to enable CPU, to validate the contents programmed into the LUTs. Checksum calculation is based on a simple bitwise addition. There are two modes in which the checksum calculation works.

- Checksum mode
- MISR (Multi Input Shift Register) mode

In checksum mode, carry on the MSB ([0] in this case) is dropped. In MISR mode, carry on MSB is added back to the LSB ([31]).

Please see the following example in the figure to see how checksum is calculated in checksum mode on every write on the LUT:

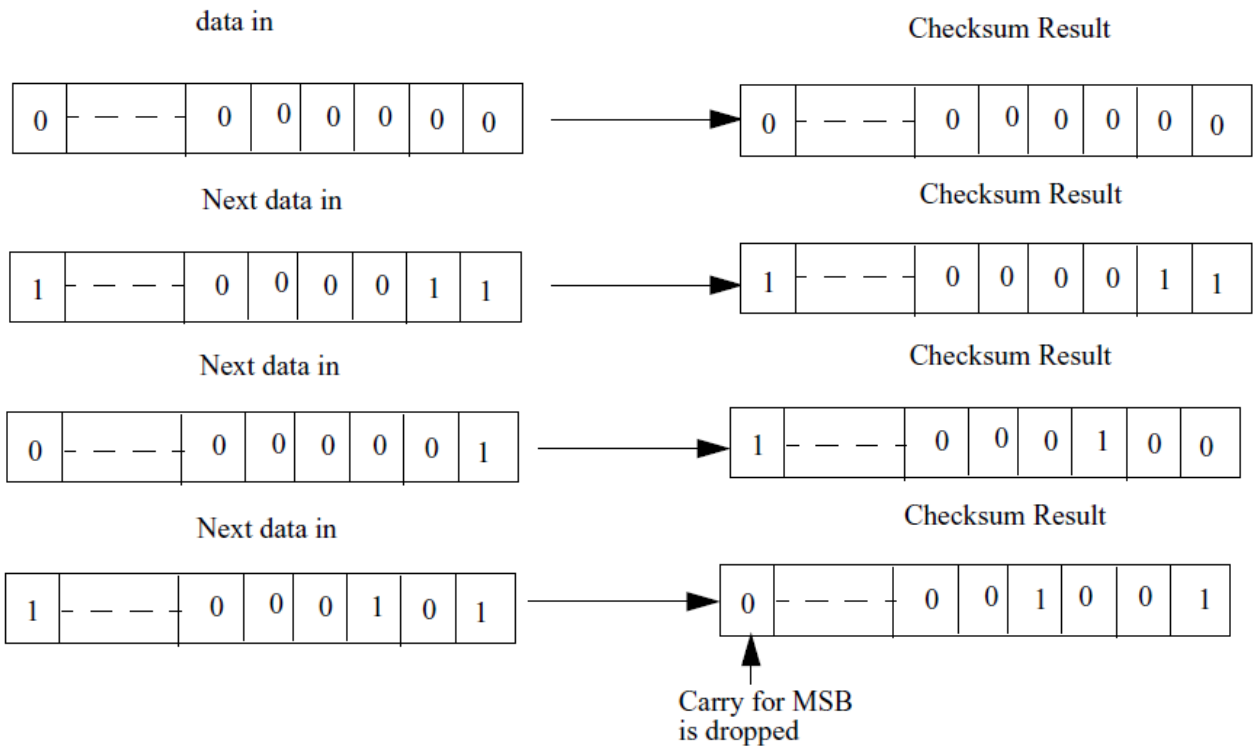


Figure 47-17. Checksum Calculation Example

47.9 Lookup Table Configuration

The following two figures show the examples of LUT configuration. The starting location depends upon the value of the WGM Address register, which is used to load an internal address counter which then increments on every write to the LUT data register. There are 2 modes in which the LUT can be configured.

- 32-bit ([0:11] are for 12-bit DAC and [12:15] are unused)
- 16-bit ([16:27] are for 12-bit DAC and [28:31] are unused)

Following is the example with 32-bit write:

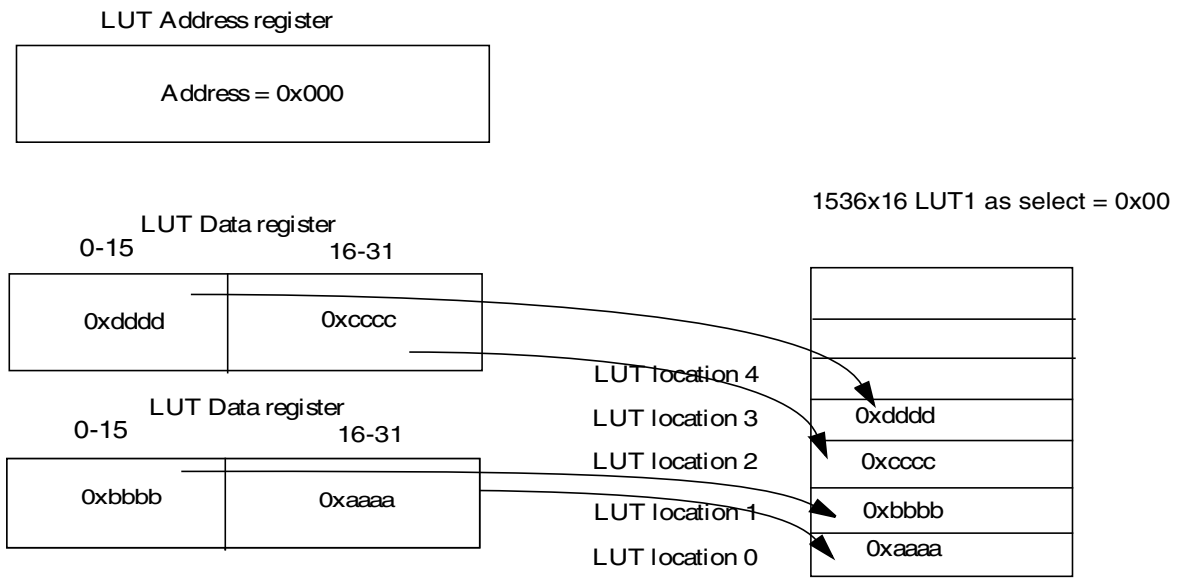


Figure 47-18. 32-bit LUT Configuration Example

Following is the example with 16-bit write:

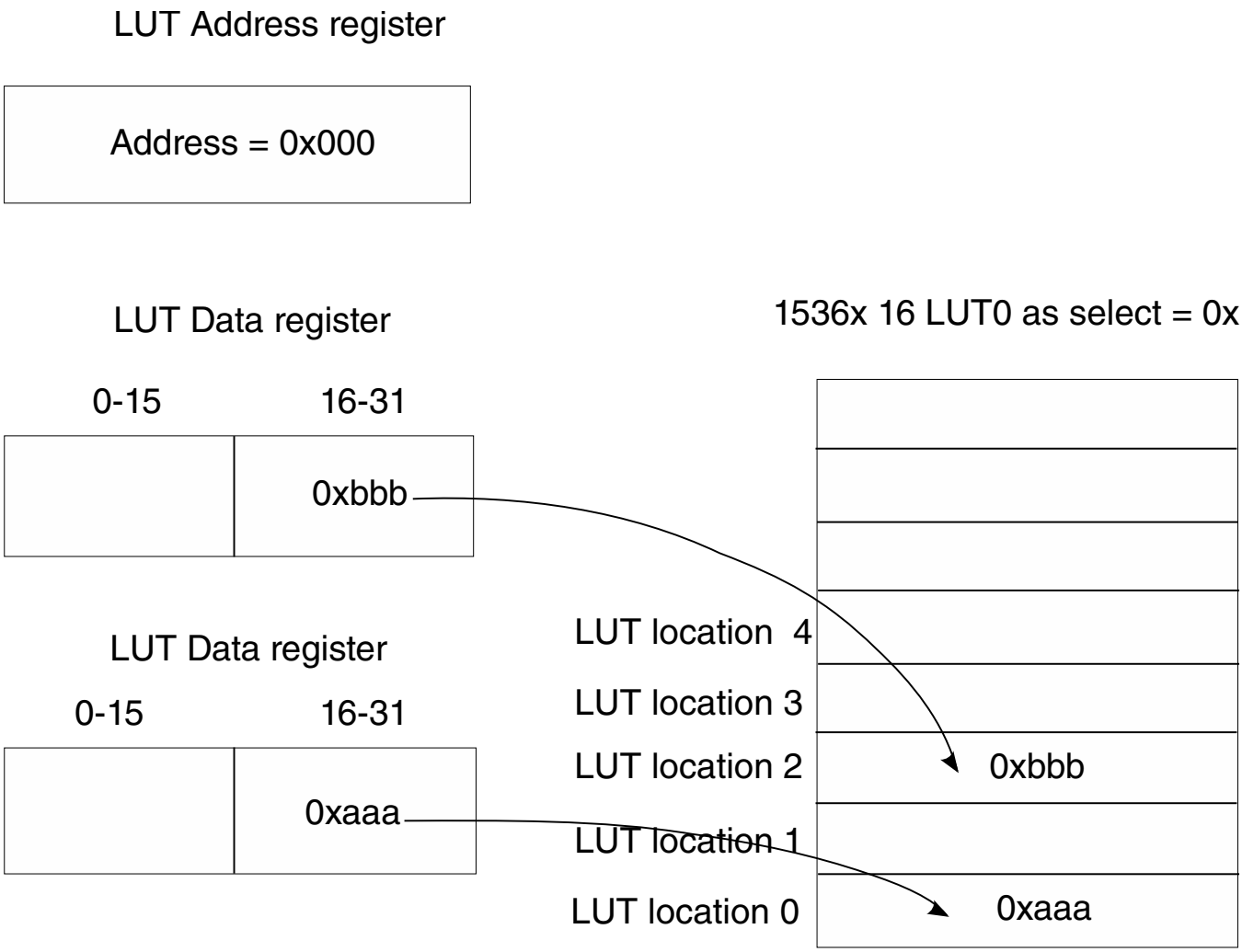


Figure 47-19. 16-bit LUT Configuration Example

47.10 WGM Initialization Sequence

Following section describes how various modules/interfaces of WGM should be configured in order to achieve the required functionality.

47.10.1 WGM-DAC Initialization Sequence

For programming the LUTs, CPU will need to take the following steps:

- Check that DEBUG[CONT_ERR] is low.
- Configure the LUT_ADDR. Current depth of LUT is 768 meaning that WGM_LUT_ADDR [22:31] will be used for addressing. WGM_LUT_ADDR [20:21] will select among the 4 available LUTs.

- Write the data to be written in the LUT_DATA [0:31]. This causes the logic to transfer the data contents [16:31] into the selected LUT location as per the address register content and data contents [0:15] to the next LUT location. Logic then uploads an internal counter with the address register content. This counter increments by 1 on every write on the WGM_LUT_DATA. Please note that the CPU write enable signal is used to increment the counter. This will ensure that the next write to the data register will transfer the contents into the next LUT location. The counter will roll over once it reaches 767. This is how the entire LUT or a selected portion of the LUT is configured.
- Verify if the data contents were written properly by comparing the checksum result with WGM_LUT_CKSM.
- Configure the CTRL([CLENR] of CTRL register) value so as to match the DAC capacity.
- Configure the CTRL([CSTEP] of CTRL register. The maximum data rate supported in 10 Msps. User should configure in a way that the maximum data rate does not exceed 10 Msps.
- Configure CTRL1[EDMA_CTL] in CTRL_1 register.
- Configure VALD_CNT on the basis of how many entries in a LUT is to be used to send data to the DAC.
- Configure CVAL register.
- Configure FADR_VAL in CTRL_1 register.
- Configure WGM_INT_EN [31] and WGM_INT_EN [30] for address counter start and end interrupts in INT_EN register.
- Configure WGM Control Register 1 (WGM_CTRL_1) [WGM_MODE]. This should be set to 1 for stand alone mode operation.
- Enable WGM by setting of WGM Control Register 1 (WGM_CTRL_1) [WGM_EN].

47.10.2 LFSR 128/360 Initialization Sequence

Following are the steps which are required to configure the LFSR modules.

- Configure "lfsr_sel" in LFSR Select Register to decide which LFSR module out of the five, needs to be configured.
- Configure "lfsr_dw_sel" in LFSR Select Register to select which 32 bit chunk needs to be updated.
- Configure first 32 bits of seed in WGM LFSR128 Register.

Chapter 48
RADAR Analog Front-End (AFE)

48.1 Chip specific AFE information

48.1.1 AFE filter phase select logic

It is possible to trigger the AFE filter logic based to sequence the AFE filters in a sequential order. This triggering mechanism can be used in two modes as shown in the figure below:

- Trigger after a wait cycle(recommended mode)
- Trigger without a wait state

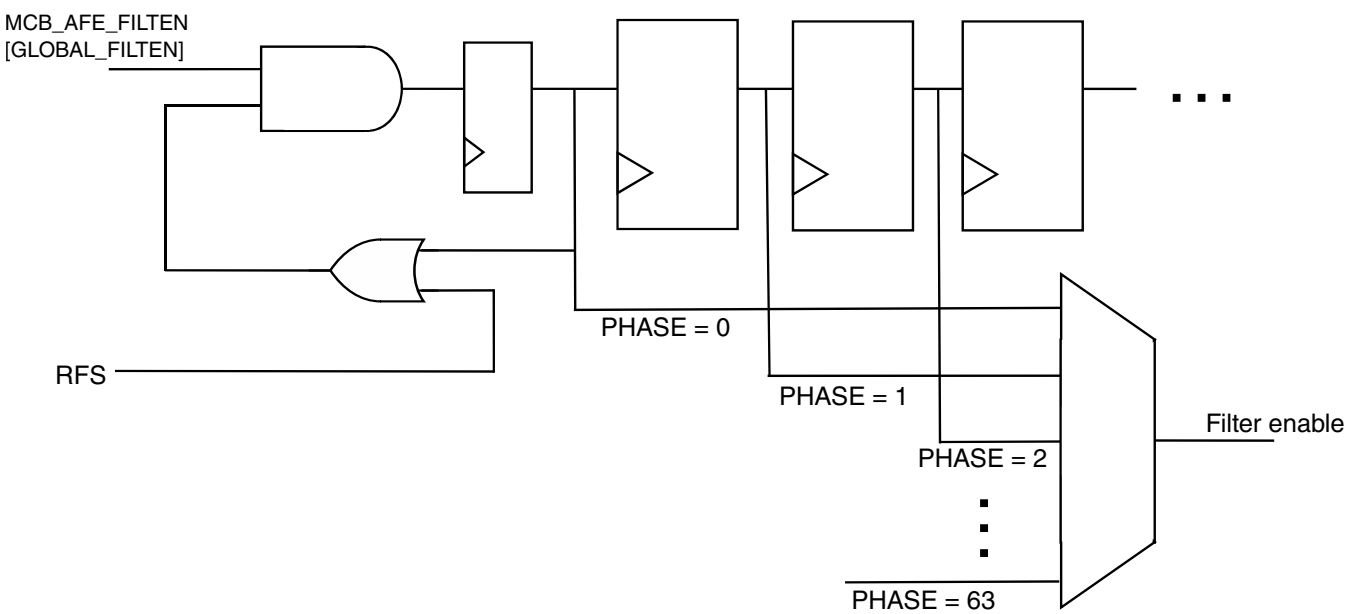


Figure 48-1. Decimation filter phase delay with MCB_AFE_FILTn_PHASE[MODE] = 0

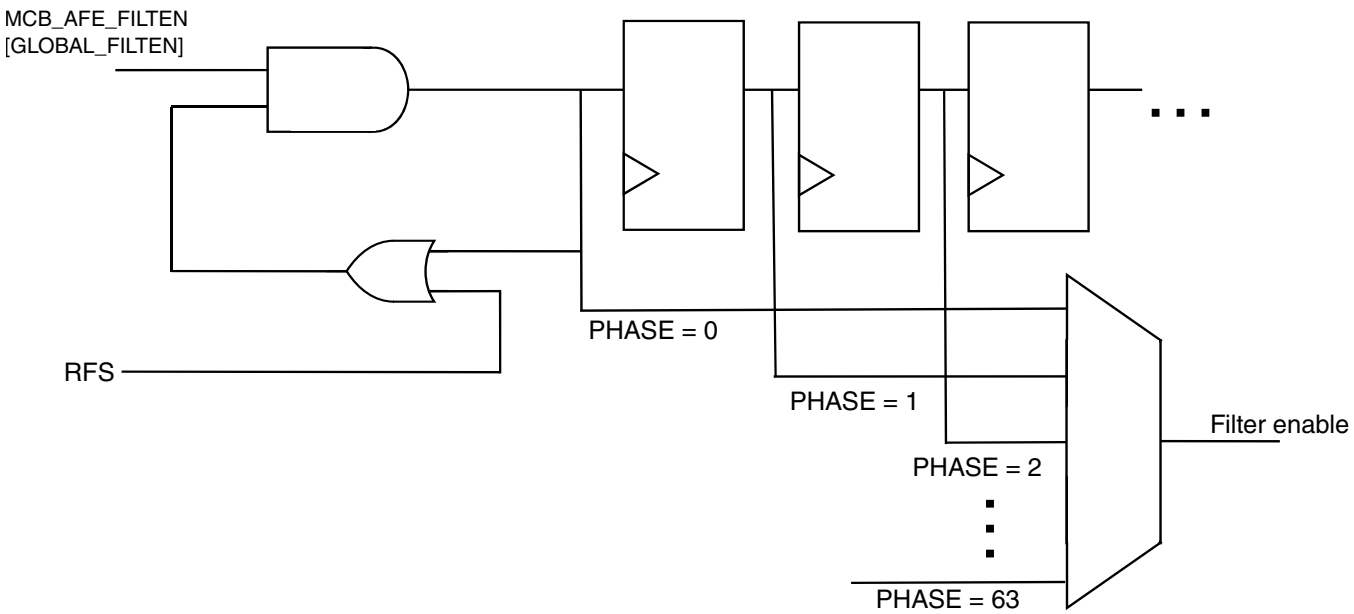


Figure 48-2. Decimation filter phase delay with MCB_AFE_FILTn_PHASE[MODE] = 1

- The Global Filter enable bit shown in the diagram above is controlled by MCB_AFE_FILTEN[GLOBAL_FILTEN].
- The Mode bit is controlled by MCB_AFE_FILT*n*_PHASE[MODE].
- The Phase is controlled by MCB_AFE_FILT*n*_PHASE[PHASE].

Using the AFE decimation filters :

Generic description for the AFE decimation filters is provided in Control and Operation .

Specifically for this device, follow the steps below to operate the AFE decimation filters:

1. Clear MCB_AFE_FILTEN to ensure decimation filters are not operating.
2. Write desired values to the AFE_FLnCOEFn registers and the AFE_FILTnCTRL[OSR] bits to configure decimation filters.
3. Set MCB_AFE_FILTEN to enable the filters once RFS is received.
4. The filters will start producing output words after the CTE module asserts the filter enables (RFS).

48.1.2 AFE filter phase select logic

To obtain clean SD-ADC data, the user needs to ensure that the minimum time between RFS, which enables the decimation filter, and assertion of acquisition window from CTE (ACQ_WIN) to be as follows:

$T_{filt\_en} + T_{phase\_mode}$ + 8 sample times

Similarly the minimum time between start of stable analog input and ACQ_WIN must be as follows:

9 sample times

T_{filt_en} depends on the OSR value in AFE_FILTCTRLn. Based on 80 MHz clock:
- OSR = 00: 19 cycles
- OSR = 01: 27 cycles
- OSR = 10: 43 cycles
- OSR = 11: 75 cycles

T_{phase_mode} depends on the Filter Phase setting in MCB_AFE_FILTn_PHASE.Example based on 80 MHz clock: When MODE = 0 (one cycle delay), PHASE = 2;T_{phase_mode} = 1 + 2 = 3 cycles

48.2 RADAR Analog Front-End (AFE) Wrapper

The AFE wrapper module contains control registers for use with the AFE hard block. The wrapper also contains decimation filters for the outputs of the 4 ADC's in the AFE hard block. The filter results are fed to the Signal Processing Toolbox (SPT) module. The control registers use a peripheral bus interface to communicate with the device. The AFE generates 320 MHz, 160 MHz, and 80 MHz clocks from the 320 MHz SDPLL output.

48.2.1 Features

The AFE analog block has the following features:
- Generates 320 MHz, 160 MHz, and 80 MHz clocks using the SDPLL output
- Four continuous-time sigma delta ADCs
- The AFE provides differential inputs for four separate high dynamic range sigma delta modulators with two poles of built in anti-aliasing filter performance and 5 MHz bandwidths to allow for multichannel simultaneous processing of sensitive analog signals
- The AFE provides high accuracy calibrated 12-bit DAC with accurate rise and fall times for improved accuracy and low step error offsets
- Provides low jitter and low phase noise 320 MHz clock as well as a low jitter 40 MHz crystal reference clock
- Tuning and mismatch shaping algorithms are part of the analog hard block. These algorithms provide feedback data for tuning the ADC coefficients and improving the ADC performance
- Sigma delta ADCs each have separate mismatch shaping algorithms for improved distortion performance

- The AFE is self contained with local regulation of analog and digital supplies using a combination of on-chip and off-chip bypassing to obtain optimal supply rejection and isolation performance
- The AFE provides for accurate trimmed bias currents as well as a trimmed bandgap voltage
- The AFE provides for bypassing the 40 MHz crystal reference clock

The AFE wrapper has the following features:

- Generates phase indicator to maintain alignment between the DAC and WGM
- Generates an interrupt based on ADC input overvoltage condition
- Provides decimation filtering of Sigma-Delta ADC outputs
- Individual power down controls for the SDADCs and DAC. Power down of the XOSC and SDPLL is controlled through the Mode Entry Module Mode Configuration registers.

48.2.2 Modes of Operation

The AFE has many modes of operation to enable proper processing of signals with high dynamic range accurately. The various modes of operation also include various test modes for testing DC bias conditions within the AFE to ensure proper operation of the various analog processing blocks. The AFE includes modes to calibrate the high precision analog blocks as well as one bypass mode of operation.

- Run Mode

  XOSC and SDPLL are enabled. SDPLL is locked at 320MHz.
  Up to four channels of the ADC are enabled and ready to process signals.
  The 12-bit DAC is ready to process data from the main processor in the IC.

- Calibration Mode (enabled during initial testing and possibly during a given period of inactivity for the AFE).

  In calibration mode, a trim comparator is enabled to accurately trim to an external resistor. The ADC uses the trim codes provided by the trim comparator to tune the resistor array within the ADC. The ADC sends corrected trim codes for the ADC coefficients in this mode of operation.

- SDPLL Calibration Mode

  SDPLL must run calibration routine to lock at 320 MHz.

- XOSC Bypass Mode

In this mode, the crystal oscillator can be bypassed allowing a single-ended external 40MHz clock to be supplied via the extal input pad or a differential external 40MHz clock to be supplied via the extal and xtal input pads.

- Power down Modes

  The Power down modes enable power down of individual modules and sub-blocks.

48.2.3 AFE Wrapper Block Diagram

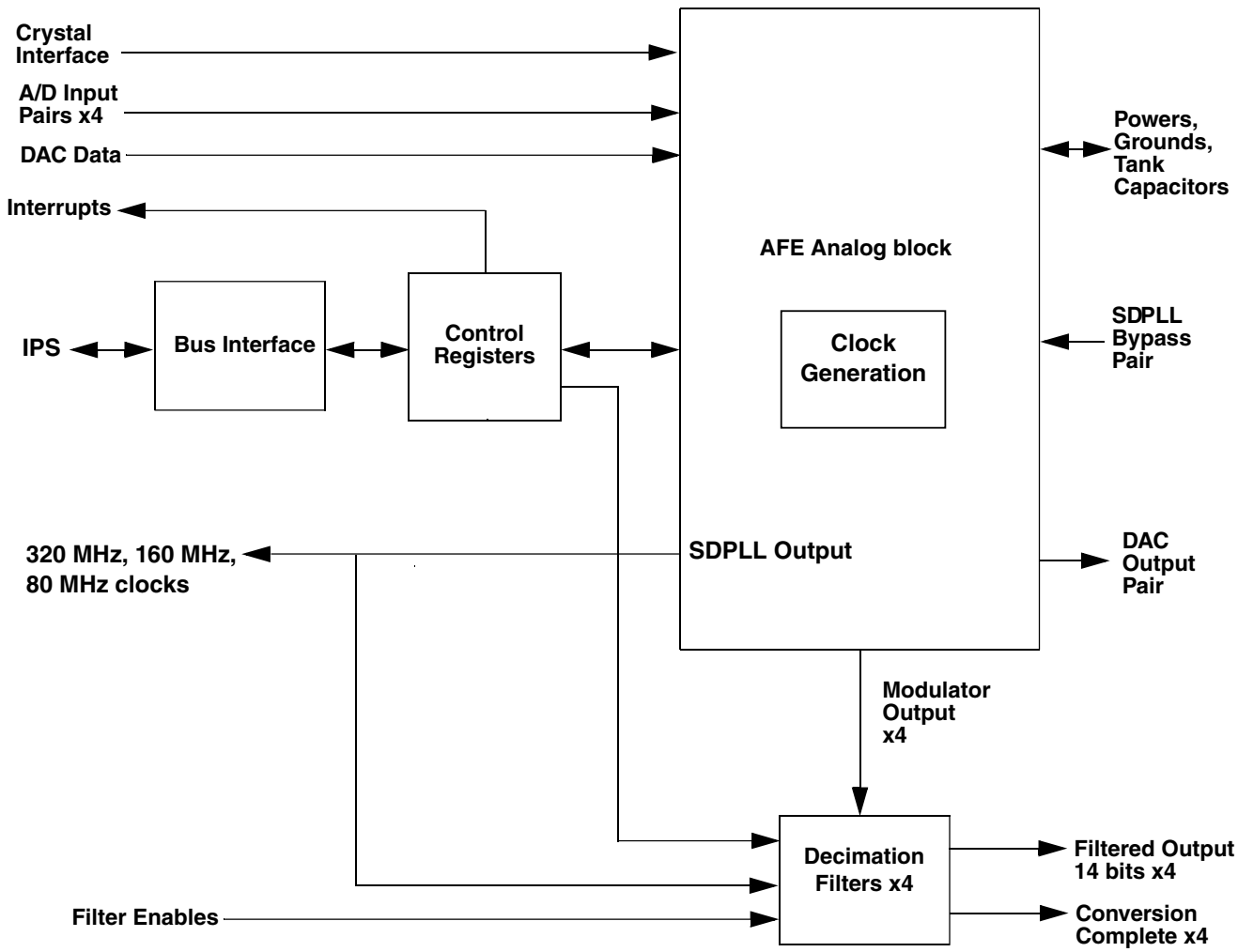


Figure 48-3. AFE Wrapper Block Diagram

48.2.4 AFE Analog Block Diagram

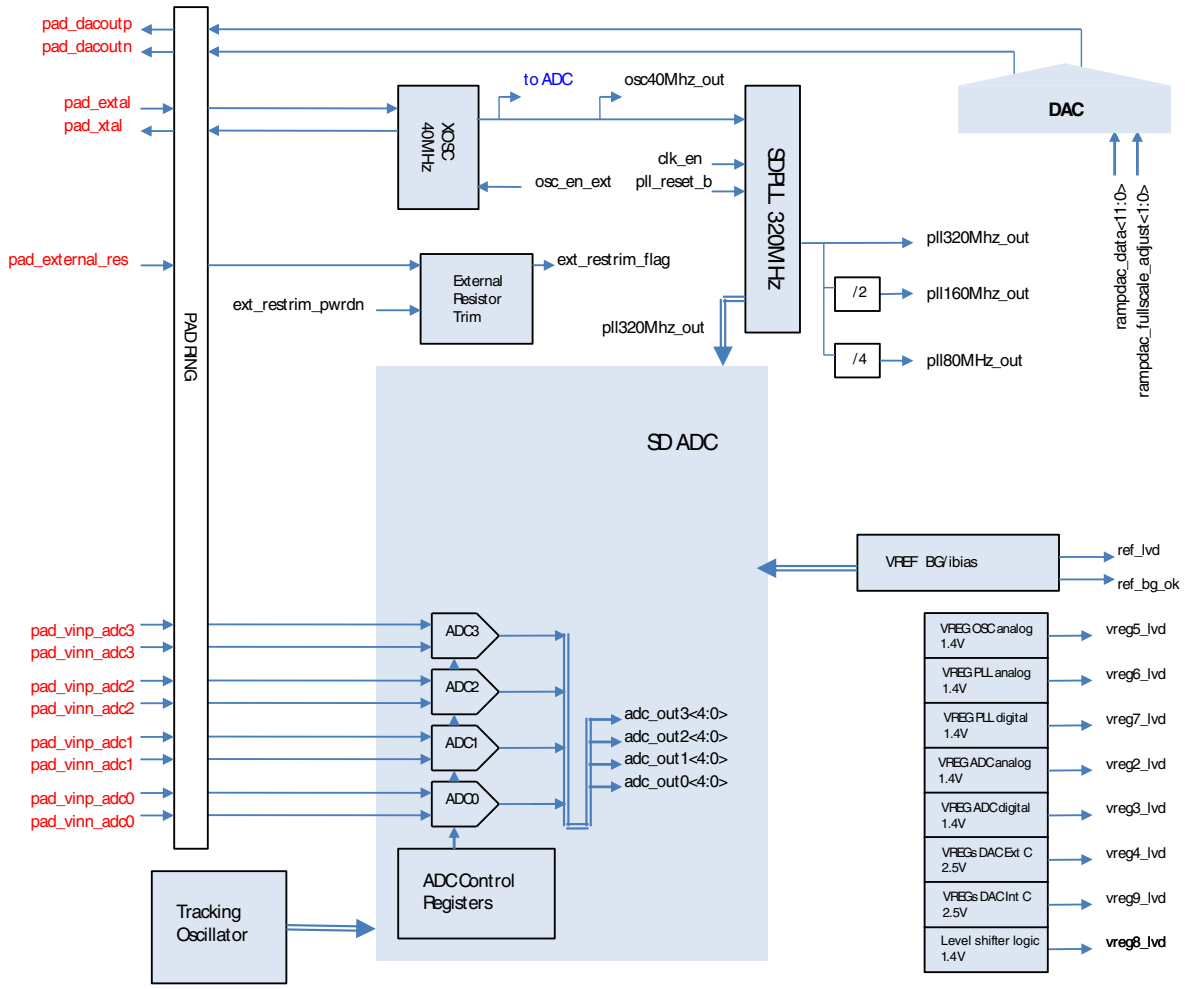


Figure 48-4. AFE Analog Block Diagram

48.2.5 Trimming

48.2.5.1 Resistor Trimming

The following steps are used to trim the external resistor.

NOTE

If no external resistor is connected to the SD_R pin, then this procedure must not be executed and the SDADCs and the DAC cannot be used.

1. Clear ADCTRIM[RMASK].

2. Clear ADCCTRL2[RT_PWRDN].
3. Wait for ADCTRIM[RESRDY] to go high.
4. Set ADCCTRL2[RT_PWRDN] to turn off the external resistor trim block.

48.2.5.2 Capacitor Trimming

The following steps are used to trim the tracking oscillator.

1. Perform resistor trimming. Without resistor trimming capacitor trimming will not work.
2. Write the value 011 to ADCTRIM[OVRD_CAP] and then set ADCTRIM[COVRD].

48.2.6 Initialization

When exiting reset, use the following procedure to turn on the AFE and start processing. If not starting from a reset condition, make sure all registers are set to their reset values before starting this procedure.

VREF/IREF and VREG8 will power-on automatically on power supply (VDD_HV_RAW and VDD_HV_DAC) ramp.

Voltage regulators other than VREG8 will power-on automatically on power supply ramp (VDD_HV_RAW and VDD_HV_DAC) when VREF LVD and LVDSTS[BGOK] are both logic 1.

When the voltage regulator low voltage detects output a high (no low voltage), the other modules (such as the ADC, DAC, SDPLL, OSC, etc.) can be powered-up.

If VREF LVD or LVDSTS[BGOK] are logic 0, all voltage regulators (except VREG8) are automatically powered down and their LVD's output a logic 0.

Start initialization:

1. Clear VRFCTRL1[PWRDN] to make sure Vref is on
2. Wait for LVDSTS[BGOK] to be set to make sure Vref is present

Steps to turn on oscillator:

If using a crystal:

1. Clear OSCCTRL[EN_EXT].
2. Set OSCCTRL[ICAL] to 0110.

3. Write the desired oscillator settling time into OSCDLY[EOCV]. (Reset value is 0x80 and is recommended.)
4. Enable the oscillator in the Mode Entry module.
5. Wait for OSCSTS[STS] to get set.
6. Write OSCSTS[STS] to clear interrupt.

If using single-ended bypass mode:

1. Set MCB_MISC1[SIN_END_BYP].
2. Set OSCCTRL[EN_EXT].
3. Set OSCCTRL[ICAL] to 0110.
4. Write the desired oscillator settling time into OSCDLY[EOCV]. (Reset value is 0x80 and is recommended.)
5. Set MC_ME_mode_MC[XOSCON] in the Mode Entry module and perform a mode transition.
6. Wait for OSCSTS[STS] to get set.
7. Write OSCSTS[STS] to clear interrupt.

If using differential bypass mode:

1. Set OSCCTRL[EN_EXT] and clear OSCCTRL[31] in one register write command.
2. Set OSCCTRL[ICAL] to 0110.
3. Write the desired oscillator settling time into OSCDLY[EOCV]. (Reset value is 0x80 and is recommended.)
4. Set MC_ME_mode_MC[XOSCON] in the Mode Entry module and perform a mode transition.
5. Wait for OSCSTS[STS] to get set.
6. Write OSCSTS[STS] to clear interrupt.

Steps to turn on and calibrate SDPLL:

1. Clear MC_ME_(D)RUNn_MC[SDPLLON] in Mode Entry Module and perform mode change in order to make sure SDPLL is reset and off.
2. Clear PLLCTRL1[RST_B] in AFE.
3. Set PLLCTRL1[CLKGEN_EN].
4. Write 0x7 to PLLCTRL1[DCBIAS_HI_LIM] and 0x4 to PLLCTRL1[DCBIAS_LO_LIM].
5. Write 0x3f to PLLCTRL2[FCAP_HI_LIM] and 0x00 to PLLCTRL2[FCAP_LO_LIM].
6. Set PLLCTRL8[REFCLK_CNT] to 0xF and set PLLCTRL8[FCLKBY2_CNT] to 0x0F0.
7. Set PLLCTRL3[CP_I_SEL] to 100.

8. Turn on SDPLL using the Mode Entry module and perform a mode change (do not yet check for mode change completion as next steps are required before mode change can complete).
9. Set PLLCTRL1[RST_B] to take SDPLL out of reset.
10. Enable SDPLL calibration by setting PLLCTRL2[START].
11. Wait for PLLSTS[LOCK] to get set which indicates completion of calibration (500µsec) and lock (75µsec).
12. If desired set PLLCTRL3[LCKLOIE] to enable loss of lock interrupt.
13. If desired set PLLCTRL3[LORIE] to enable loss of reference interrupt.
14. Check for mode change completion.

Steps to turn on analog circuits:

1. Perform resistor array trimming as described in Resistor Trimming section and Capacitor array trimming as described in Capacitor Trimming section.
2. Turn on ADC protection circuitry by clearing and ADCCTRL7[OVPWRDN]. Wait 3 microseconds before the next step.
3. Turn on the ADC's by clearing the ADCCTRL7[PWRDN] bits for the ADC channel's that are to be used. It is recommended to not power up all ADC's simultaneously.
4. Set ADCCTRL1[MM_SEL] to 10.
5. Set ADCCTRL1[MM_DTHEN] to 0011.
6. Set ADCCTRL1[MM_BYP] to 0 to select mismatch shaping or 1 to de-select mismatch shaping.
7. Clear any ADCOVLD[OVERVOLTS] or flags by writing 1's to those bit positions.
8. Turn on DAC circuitry by clearing DACCTRL[PWRDN].
9. Set DACCTRL[FS_ADJ] to 01 or 10.
10. Initialize ADC by setting ADCRST[INTRST], ADCRST[DFFRST], and ADCCTRL1[MM_RST]. Next, clear ADCRST[INTRST], then clear ADCRST[DFFRST], and then clear ADCCTRL1[MM_RST].

Steps to turn on decimation filters:

1. Clear MCB_AFE_FILTEN to make sure the decimation filters are not working.
2. Configure decimation filters by writing desired values to the FLnCOEFn registers.
3. Write desired values to FILTnCTRL[OSR] bits.
4. Write desired values to FILTnCTRL[USE14] bits.
5. Set MCB_AFE_FILTEN to enable the filters once RFS is received.
6. The filters will start producing output words after the CTE module asserts the filter enables.

48.3 Memory map and register description

Transfer errors are generated for accesses beyond the largest register address but within the 16 KB block of addresses reserved for the AFE module.

NOTE

AFE does not support transfer error for memory holes.

AFE memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Oscillator Control Register (AFE_OSCCTRL)	32	R/W	0000_00A1h	48.3.1/2206
4	Oscillator Status Register (AFE_OSCSTS)	32	R	0000_0000h	48.3.2/2208
8	Oscillator Delay Register (AFE_OSCDLY)	32	R/W	0000_0080h	48.3.3/2209
C	SDPLL Control Register 1 (AFE_PLLCTRL1)	32	R/W	00F0_0001h	48.3.4/2210
10	SDPLL Control Register 2 (AFE_PLLCTRL2)	32	R/W	3F00_0000h	48.3.5/2212
14	SDPLL Control Register 3 (AFE_PLLCTRL3)	32	R/W	0000_0000h	48.3.6/2214
28	SDPLL Control Register 8 (AFE_PLLCTRL8)	32	R/W	0000_0000h	48.3.7/2216
2C	SDPLL Status Register (AFE_PLLSTS)	32	R	0000_0002h	48.3.8/2217
30	ADC Control Register 1 (AFE_ADCCTRL1)	32	R/W	0000_0001h	48.3.9/2219
34	ADC Control Register 2 (AFE_ADCCTRL2)	32	R/W	0000_0201h	48.3.10/ 2221
38	ADC Tracking Oscillator Trim Register (AFE_ADCTOT)	32	R/W	0000_0000h	48.3.11/ 2222
3C	ADC Reset Register (AFE_ADCRST)	32	R/W	0000_0000h	48.3.12/ 2223
40	ADC Trim Register (AFE_ADCTRIM)	32	R/W	0100_8000h	48.3.13/ 2224
48	ADC Control Register 7 (AFE_ADCCTRL7)	32	R/W	0000_40FFh	48.3.14/ 2226
4C	ADC Overload Detect Register (AFE_ADCOVLD)	32	R/W	0000_0000h	48.3.15/ 2227
50	DAC Control Register (AFE_DACCTRL)	32	R/W	0000_0001h	48.3.16/ 2228
54	VREF Control Register 1 (AFE_VRFCTRL1)	32	R/W	0000_0000h	48.3.17/ 2230
5C	Low Voltage Detect Status Register (AFE_LVDSTS)	32	R/W	0200_0000h	48.3.18/ 2231
60	VREG Reserved Register (AFE_VRGRSVD)	32	R (reads 0)	0000_0000h	48.3.19/ 2232
64	VREG2 Control Register (AFE_VRGCTRL2)	32	R/W	0000_0000h	48.3.20/ 2233
68	VREG3 Control Register (AFE_VRGCTRL3)	32	R/W	0000_0000h	48.3.21/ 2234

Table continues on the next page...

AFE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6C	VREG4 Control Register (AFE_VRGCTRL4)	32	R/W	0000_0000h	48.3.22/ 2236
70	VREG5 Control Register (AFE_VRGCTRL5)	32	R/W	0000_0000h	48.3.23/ 2237
74	VREG6 Control Register (AFE_VRGCTRL6)	32	R/W	0000_0000h	48.3.24/ 2239
78	VREG7 Control Register (AFE_VRGCTRL7)	32	R/W	0000_0000h	48.3.25/ 2240
7C	VREG8 Control Register (AFE_VRGCTRL8)	32	R/W	0000_0000h	48.3.26/ 2242
84	VREG9 Control Register (AFE_VRGCTRL9)	32	R/W	0000_0000h	48.3.27/ 2243
A0	Decimation Filter Control Register (AFE_FILTCTRL0)	32	R/W	0000_0000h	48.3.28/ 2245
A4	Decimation Filter Control Register (AFE_FILTCTRL1)	32	R/W	0000_0000h	48.3.28/ 2245
A8	Decimation Filter Control Register (AFE_FILTCTRL2)	32	R/W	0000_0000h	48.3.28/ 2245
AC	Decimation Filter Control Register (AFE_FILTCTRL3)	32	R/W	0000_0000h	48.3.28/ 2245
C0	Decimation Filter Coefficient Register 0 (AFE_FLCOEF00)	32	R/W	0000_3FE9h	48.3.29/ 2246
C4	Decimation Filter Coefficient Register 1 (AFE_FLCOEF10)	32	R/W	0000_3FCFh	48.3.30/ 2247
C8	Decimation Filter Coefficient Register 2 (AFE_FLCOEF20)	32	R/W	0000_3FC2h	48.3.31/ 2247
CC	Decimation Filter Coefficient Register 3 (AFE_FLCOEF30)	32	R/W	0000_3FE1h	48.3.32/ 2248
D0	Decimation Filter Coefficient Register 4 (AFE_FLCOEF40)	32	R/W	0000_003Ah	48.3.33/ 2248
D4	Decimation Filter Coefficient Register 5 (AFE_FLCOEF50)	32	R/W	0000_00ACh	48.3.34/ 2249
D8	Decimation Filter Coefficient Register 6 (AFE_FLCOEF60)	32	R/W	0000_00ECh	48.3.35/ 2249
DC	Decimation Filter Coefficient Register 7 (AFE_FLCOEF70)	32	R/W	0000_00A8h	48.3.36/ 2250
E0	Decimation Filter Coefficient Register 8 (AFE_FLCOEF80)	32	R/W	0000_3FC8h	48.3.37/ 2250
E4	Decimation Filter Coefficient Register 9 (AFE_FLCOEF90)	32	R/W	0000_3EA6h	48.3.38/ 2251
E8	Decimation Filter Coefficient Register 10 (AFE_FLCOEF100)	32	R/W	0000_3E00h	48.3.39/ 2251
EC	Decimation Filter Coefficient Register 11 (AFE_FLCOEF110)	32	R/W	0000_3EA1h	48.3.40/ 2252

Table continues on the next page...

AFE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F0	Decimation Filter Coefficient Register 12 (AFE_FLCOEF120)	32	R/W	0000_00E3h	48.3.41/ 2252
F4	Decimation Filter Coefficient Register 13 (AFE_FLCOEF130)	32	R/W	0000_045Bh	48.3.42/ 2253
F8	Decimation Filter Coefficient Register 14 (AFE_FLCOEF140)	32	R/W	0000_07E5h	48.3.43/ 2253
FC	Decimation Filter Coefficient Register 15 (AFE_FLCOEF150)	32	R/W	0000_0A20h	48.3.44/ 2254
100	Decimation Filter Coefficient Register 0 (AFE_FLCOEF01)	32	R/W	0000_3FE9h	48.3.29/ 2246
104	Decimation Filter Coefficient Register 1 (AFE_FLCOEF11)	32	R/W	0000_3FCFh	48.3.30/ 2247
108	Decimation Filter Coefficient Register 2 (AFE_FLCOEF21)	32	R/W	0000_3FC2h	48.3.31/ 2247
10C	Decimation Filter Coefficient Register 3 (AFE_FLCOEF31)	32	R/W	0000_3FE1h	48.3.32/ 2248
110	Decimation Filter Coefficient Register 4 (AFE_FLCOEF41)	32	R/W	0000_003Ah	48.3.33/ 2248
114	Decimation Filter Coefficient Register 5 (AFE_FLCOEF51)	32	R/W	0000_00ACh	48.3.34/ 2249
118	Decimation Filter Coefficient Register 6 (AFE_FLCOEF61)	32	R/W	0000_00ECh	48.3.35/ 2249
11C	Decimation Filter Coefficient Register 7 (AFE_FLCOEF71)	32	R/W	0000_00A8h	48.3.36/ 2250
120	Decimation Filter Coefficient Register 8 (AFE_FLCOEF81)	32	R/W	0000_3FC8h	48.3.37/ 2250
124	Decimation Filter Coefficient Register 9 (AFE_FLCOEF91)	32	R/W	0000_3EA6h	48.3.38/ 2251
128	Decimation Filter Coefficient Register 10 (AFE_FLCOEF101)	32	R/W	0000_3E00h	48.3.39/ 2251
12C	Decimation Filter Coefficient Register 11 (AFE_FLCOEF111)	32	R/W	0000_3EA1h	48.3.40/ 2252
130	Decimation Filter Coefficient Register 12 (AFE_FLCOEF121)	32	R/W	0000_00E3h	48.3.41/ 2252
134	Decimation Filter Coefficient Register 13 (AFE_FLCOEF131)	32	R/W	0000_045Bh	48.3.42/ 2253
138	Decimation Filter Coefficient Register 14 (AFE_FLCOEF141)	32	R/W	0000_07E5h	48.3.43/ 2253
13C	Decimation Filter Coefficient Register 15 (AFE_FLCOEF151)	32	R/W	0000_0A20h	48.3.44/ 2254
140	Decimation Filter Coefficient Register 0 (AFE_FLCOEF02)	32	R/W	0000_3FE9h	48.3.29/ 2246
144	Decimation Filter Coefficient Register 1 (AFE_FLCOEF12)	32	R/W	0000_3FCFh	48.3.30/ 2247

Table continues on the next page...

AFE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
148	Decimation Filter Coefficient Register 2 (AFE_FLCOEF22)	32	R/W	0000_3FC2h	48.3.31/ 2247
14C	Decimation Filter Coefficient Register 3 (AFE_FLCOEF32)	32	R/W	0000_3FE1h	48.3.32/ 2248
150	Decimation Filter Coefficient Register 4 (AFE_FLCOEF42)	32	R/W	0000_003Ah	48.3.33/ 2248
154	Decimation Filter Coefficient Register 5 (AFE_FLCOEF52)	32	R/W	0000_00ACh	48.3.34/ 2249
158	Decimation Filter Coefficient Register 6 (AFE_FLCOEF62)	32	R/W	0000_00ECh	48.3.35/ 2249
15C	Decimation Filter Coefficient Register 7 (AFE_FLCOEF72)	32	R/W	0000_00A8h	48.3.36/ 2250
160	Decimation Filter Coefficient Register 8 (AFE_FLCOEF82)	32	R/W	0000_3FC8h	48.3.37/ 2250
164	Decimation Filter Coefficient Register 9 (AFE_FLCOEF92)	32	R/W	0000_3EA6h	48.3.38/ 2251
168	Decimation Filter Coefficient Register 10 (AFE_FLCOEF102)	32	R/W	0000_3E00h	48.3.39/ 2251
16C	Decimation Filter Coefficient Register 11 (AFE_FLCOEF112)	32	R/W	0000_3EA1h	48.3.40/ 2252
170	Decimation Filter Coefficient Register 12 (AFE_FLCOEF122)	32	R/W	0000_00E3h	48.3.41/ 2252
174	Decimation Filter Coefficient Register 13 (AFE_FLCOEF132)	32	R/W	0000_045Bh	48.3.42/ 2253
178	Decimation Filter Coefficient Register 14 (AFE_FLCOEF142)	32	R/W	0000_07E5h	48.3.43/ 2253
17C	Decimation Filter Coefficient Register 15 (AFE_FLCOEF152)	32	R/W	0000_0A20h	48.3.44/ 2254
180	Decimation Filter Coefficient Register 0 (AFE_FLCOEF03)	32	R/W	0000_3FE9h	48.3.29/ 2246
184	Decimation Filter Coefficient Register 1 (AFE_FLCOEF13)	32	R/W	0000_3FCFh	48.3.30/ 2247
188	Decimation Filter Coefficient Register 2 (AFE_FLCOEF23)	32	R/W	0000_3FC2h	48.3.31/ 2247
18C	Decimation Filter Coefficient Register 3 (AFE_FLCOEF33)	32	R/W	0000_3FE1h	48.3.32/ 2248
190	Decimation Filter Coefficient Register 4 (AFE_FLCOEF43)	32	R/W	0000_003Ah	48.3.33/ 2248
194	Decimation Filter Coefficient Register 5 (AFE_FLCOEF53)	32	R/W	0000_00ACh	48.3.34/ 2249
198	Decimation Filter Coefficient Register 6 (AFE_FLCOEF63)	32	R/W	0000_00ECh	48.3.35/ 2249
19C	Decimation Filter Coefficient Register 7 (AFE_FLCOEF73)	32	R/W	0000_00A8h	48.3.36/ 2250

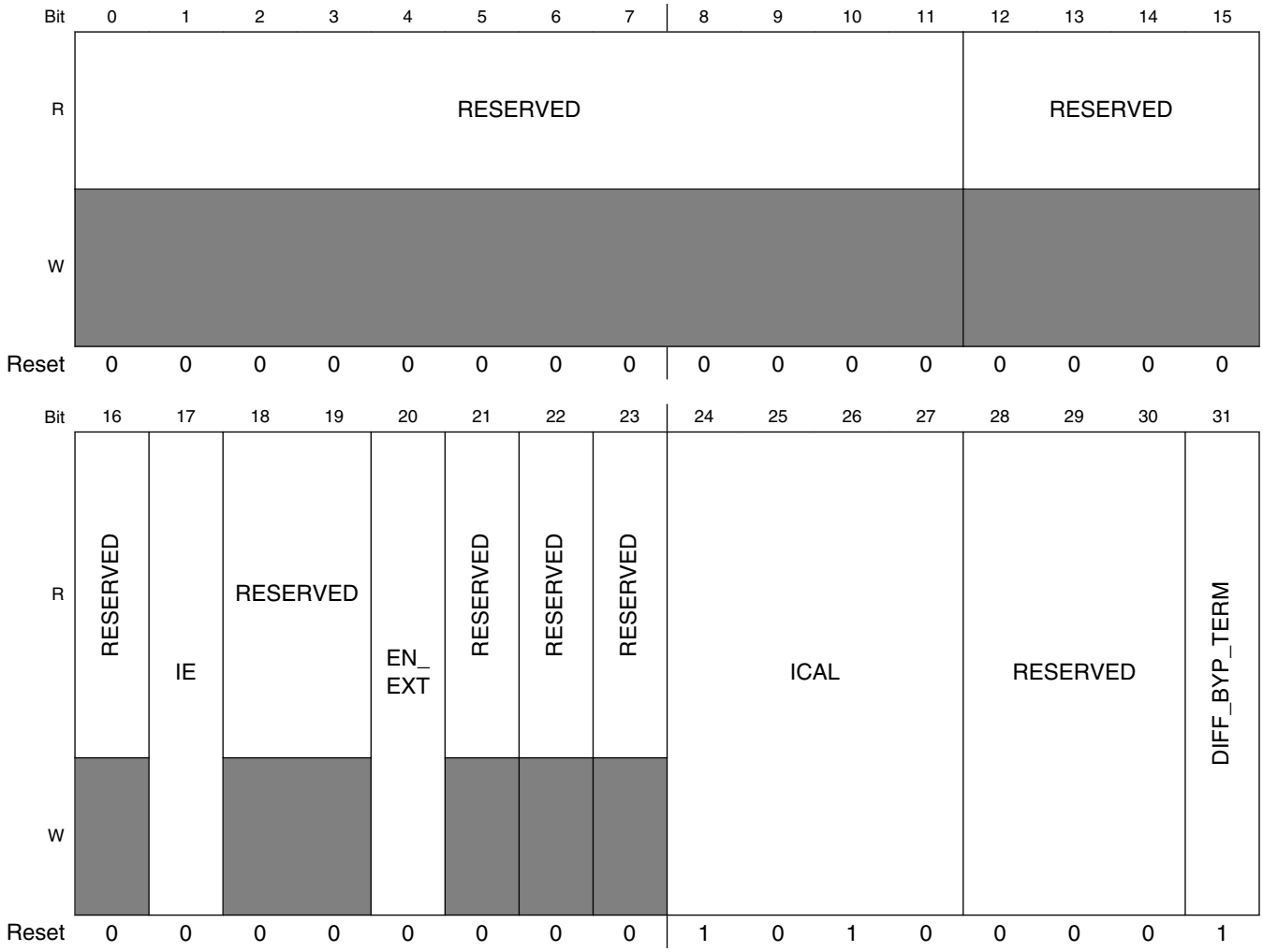
Table continues on the next page...

AFE memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A0	Decimation Filter Coefficient Register 8 (AFE_FLCOEF83)	32	R/W	0000_3FC8h	48.3.37/ 2250
1A4	Decimation Filter Coefficient Register 9 (AFE_FLCOEF93)	32	R/W	0000_3EA6h	48.3.38/ 2251
1A8	Decimation Filter Coefficient Register 10 (AFE_FLCOEF103)	32	R/W	0000_3E00h	48.3.39/ 2251
1AC	Decimation Filter Coefficient Register 11 (AFE_FLCOEF113)	32	R/W	0000_3EA1h	48.3.40/ 2252
1B0	Decimation Filter Coefficient Register 12 (AFE_FLCOEF123)	32	R/W	0000_00E3h	48.3.41/ 2252
1B4	Decimation Filter Coefficient Register 13 (AFE_FLCOEF133)	32	R/W	0000_045Bh	48.3.42/ 2253
1B8	Decimation Filter Coefficient Register 14 (AFE_FLCOEF143)	32	R/W	0000_07E5h	48.3.43/ 2253
1BC	Decimation Filter Coefficient Register 15 (AFE_FLCOEF153)	32	R/W	0000_0A20h	48.3.44/ 2254

48.3.1 Oscillator Control Register (AFE_OSCCTRL)

Address: 0h base + 0h offset = 0h



AFE_OSCCTRL field descriptions

Field	Description
0–11 RESERVED	This field is reserved.
12–15 RESERVED	This field is reserved.
16 RESERVED	This field is reserved.
17 IE	Oscillator interrupt enable. This bit allows the rising edge of the OSCSTS[STS] flag to create an interrupt. 0 Oscillator interrupt disabled. 1 Oscillator interrupt enabled.

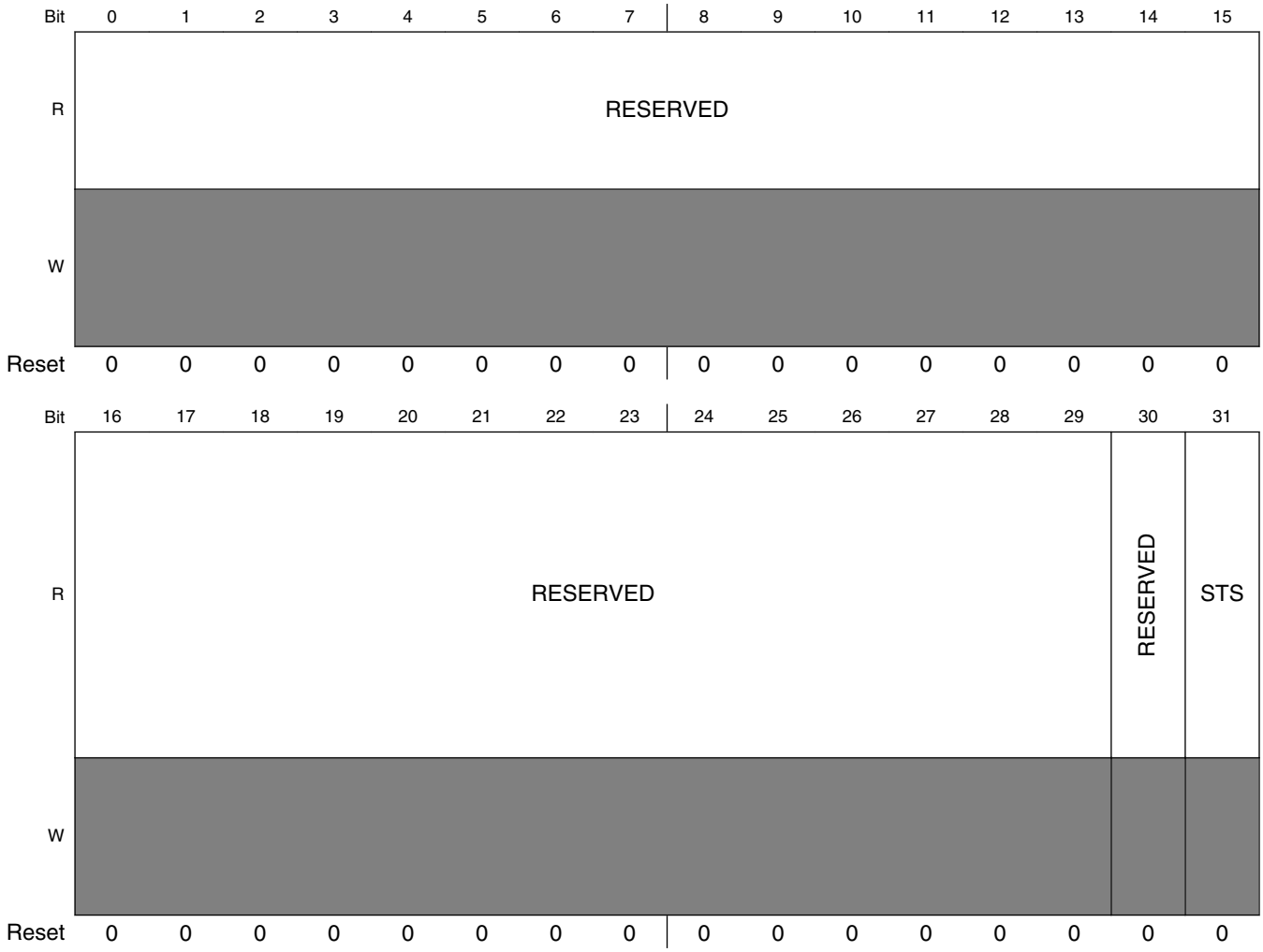
Table continues on the next page...

AFE_OSCCTRL field descriptions (continued)

Field	Description
18–19 RESERVED	This field is reserved.
20 EN_EXT	Crystal oscillator bypass control. This bit controls the crystal oscillator bypass function which allows an external 40 MHz clock to be supplied. 0 Crystal mode, no bypass. 1 40 MHz external clock enabled. Do not set this bit when the crystal oscillator is operating.
21 RESERVED	This field is reserved.
22 RESERVED	This field is reserved.
23 RESERVED	This field is reserved.
24–27 ICAL	Variable current select bits. This field controls the oscillator's and buffer's variable currents. xx00 Oscillator bias 10µA. xx01 Oscillator bias 15µA. xx10 Oscillator bias 20µA. xx11 Oscillator bias 25µA. 00xx Buffer bias 50µA. 01xx Buffer bias 100µA. 10xx Buffer bias 150µA. 11xx Buffer bias 200µA.
28–30 RESERVED	This field is reserved. This field is reserved. Always write the reset value to this field.
31 DIFF_BYP_ TERM	Termination control for differential bypass mode. Clear this field according to the procedure in the "If using differential bypass mode" instructions in Initialization. Keep it set in all other cases. 0 Termination on for differential bypass mode. 1 Termination off for all other modes.

48.3.2 Oscillator Status Register (AFE_OSCSTS)

Address: 0h base + 4h offset = 4h



AFE_OSCSTS field descriptions

Field	Description
0–29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 STS	Oscillator status This bit is set when the oscillator is powered up and enough transitions have occurred to allow an internal counter to reach a value of OSCDLY[EOCV] x 256. This bit cannot be set unless the oscillator is powered up. A rising edge on this bit will create an interrupt if OSCCTRL[IE] is set. Writing a 1 to this position will clear the oscillator interrupt but will leave this bit set. **NOTE:** STS gets cleared when XOSC is turned off using MC_ME and STS gets set when oscillator has completed transition count after being turned ON by MC_ME.

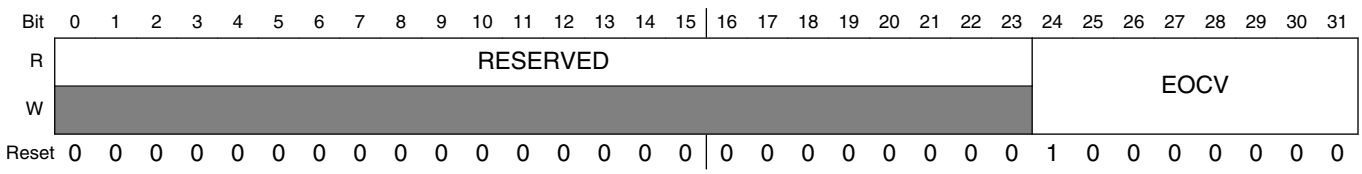
Table continues on the next page...

AFE_OSCSTS field descriptions (continued)

Field	Description
	0 Oscillator has not completed transition count. 1 Oscillator has completed transition count.

48.3.3 Oscillator Delay Register (AFE_OSCDLY)

Address: 0h base + 8h offset = 8h

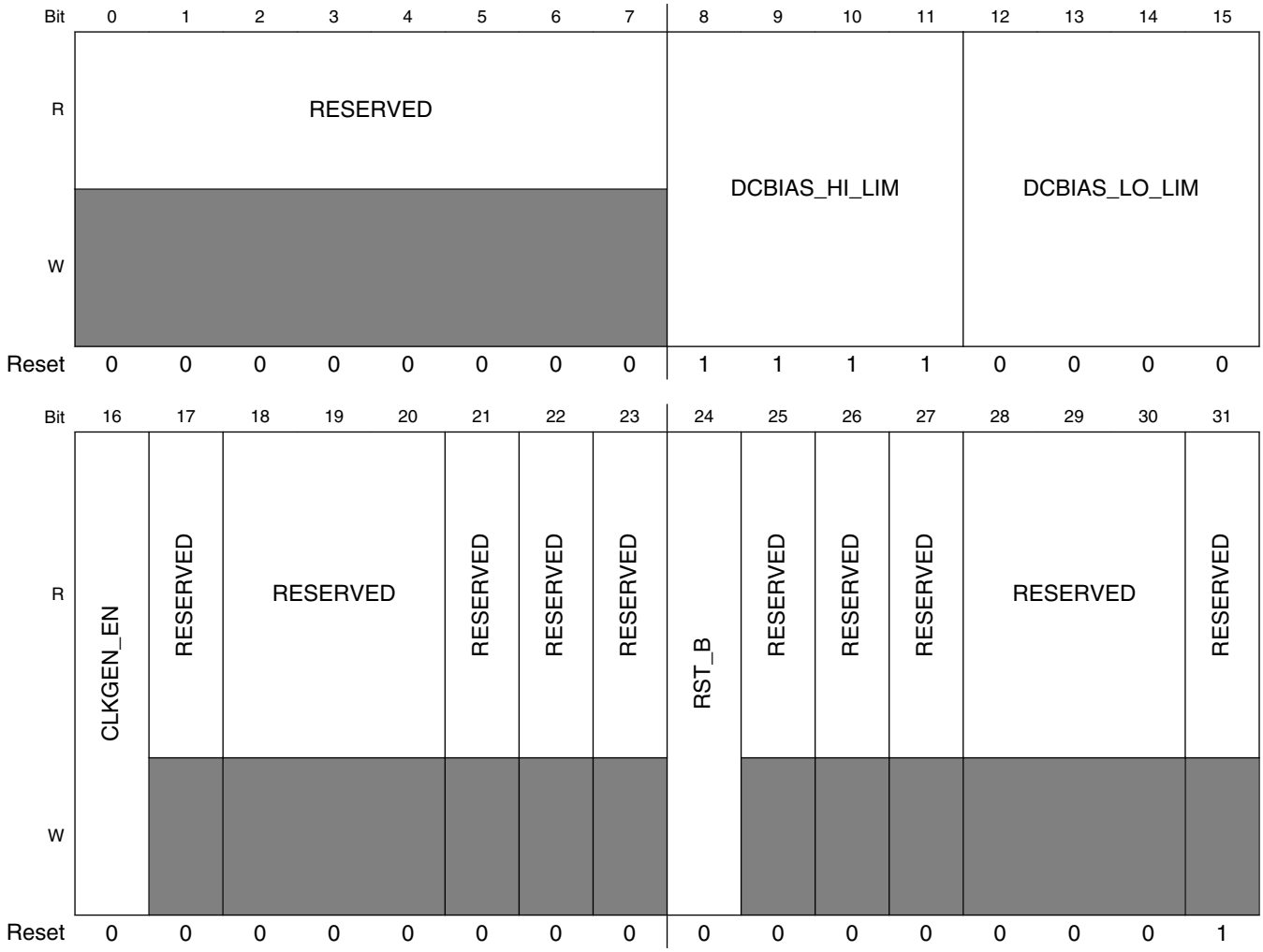


AFE_OSCDLY field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–31 EOCV	End of count value. This field specifies the end of count value to be used for comparison by the oscillator stabilization counter whenever the oscillator is switched on. This counter period ensures that the external oscillator clock signal has completed the required number of transitions before it can be selected by the system. When the oscillator counter reaches the value EOCV x 256, OSCSTS[STS] is set and may generate an interrupt. The counter is reset when the oscillator is powered down or bypassed. Changes to this register can be made before or after the oscillator is turned on (through mode entry procedure), but the user must check for OSCSTS[STS] bit after EOCV completion in both conditions after oscillator is enabled.

48.3.4 SDPLL Control Register 1 (AFE_PLLCTRL1)

Address: 0h base + Ch offset = Ch



AFE_PLLCTRL1 field descriptions

Field	Description
0–7 RESERVED	This field is reserved.
8–11 DCBIAS_HI_LIM	This field is used to limit the high (maximum) value of the DC bias bus used during the SDPLL calibration. It is used in conjunction with the DCBIAS_LO_LIM field. By default this field is 0xF and therefore does not limit the value of DC bias. This field is used during both automatic and manual calibration procedures.
12–15 DCBIAS_LO_LIM	This field is used to limit the low (minimum) value of the DC bias bus used during the SDPLL calibration. It is used in conjunction with the DCBIAS_HI_LIM field. By default this field is 0x0 and therefore does not limit the value of DC bias. This field is used during both automatic and manual calibration procedures.
16 CLKGEN_EN	Clock generation enable. This bit turns on the clock generation circuit which uses the 320 MHz output of the SDPLL to generate 160 MHz and 80 MHz clocks. This bit must be set in order for the decimation filters to operate.

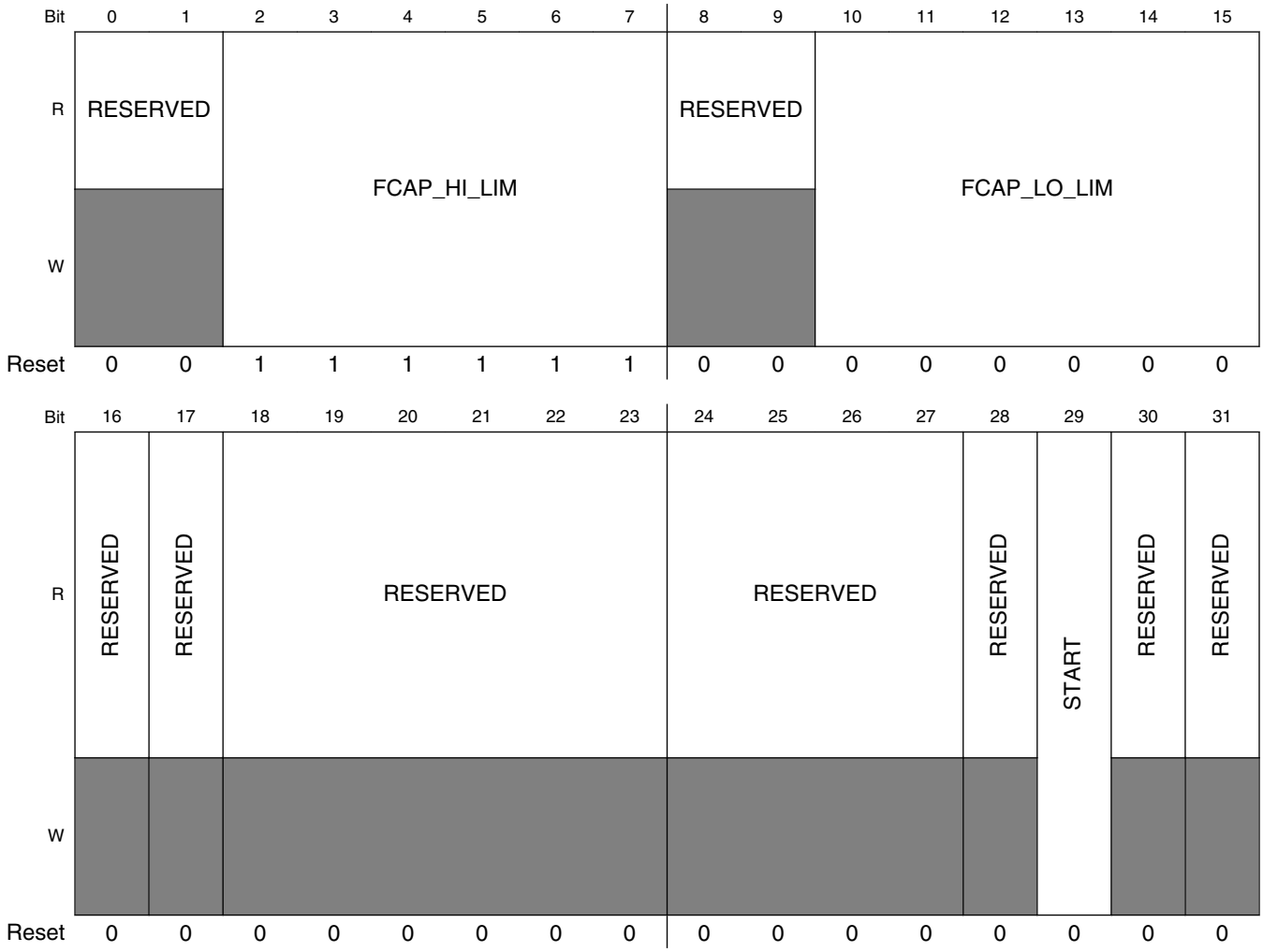
Table continues on the next page...

AFE_PLLCTRL1 field descriptions (continued)

Field	Description
	0 Disable clock generation. 1 Enable generation of 160 MHz and 80 MHz clocks.
17 RESERVED	This field is reserved.
18–20 RESERVED	This field is reserved.
21 RESERVED	This field is reserved.
22 RESERVED	This field is reserved.
23 RESERVED	This field is reserved.
24 RST_B	Lock reset. This bit resets the SDPLL lock detect and test circuitry. 0 Reset the SDPLL lock detect and test circuits. 1 Normal operation.
25 RESERVED	This field is reserved.
26 RESERVED	This field is reserved.
27 RESERVED	This field is reserved.
28–30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.5 SDPLL Control Register 2 (AFE_PLLCTRL2)

Address: 0h base + 10h offset = 10h



AFE_PLLCTRL2 field descriptions

Field	Description
0–1 RESERVED	This field is reserved.
2–7 FCAP_HI_LIM	This field is used to limit the high (maximum) value of the FCAP bus used during the SDPLL calibration. It is used in conjunction with the FCAP_LO_LIM field. By default this field is 0x3F and therefore does not limit the value of FCAP. This field is used during both automatic and manual calibration procedures.
8–9 RESERVED	This field is reserved.
10–15 FCAP_LO_LIM	This field is used to limit the low (minimum) value of the FCAP bus used during the SDPLL calibration. It is used in conjunction with the FCAP_HI_LIM field. By default this field is 0x00 and therefore does not limit the value of FCAP. This field is used during both automatic and manual calibration procedures.

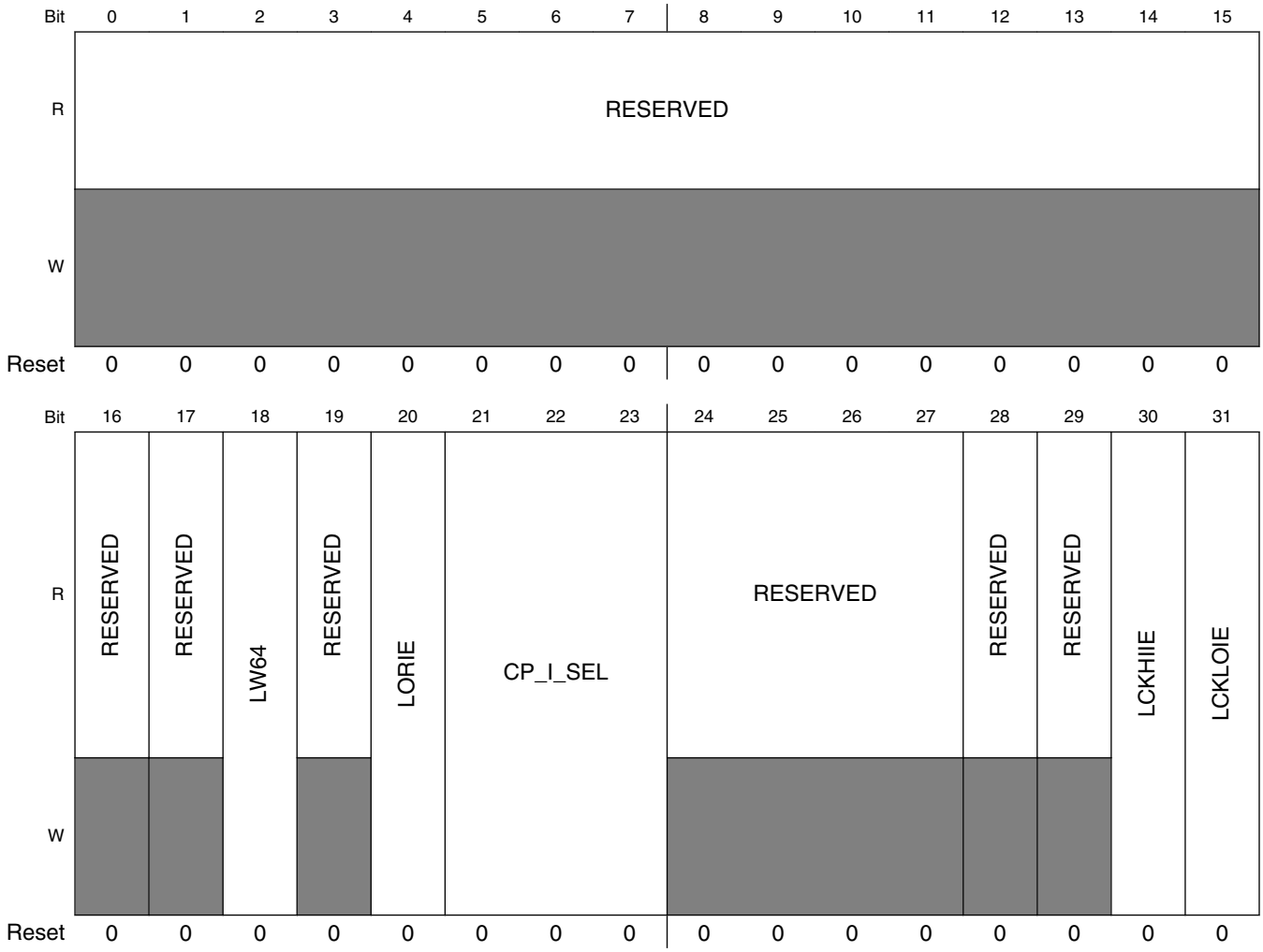
Table continues on the next page...

AFE_PLLCTRL2 field descriptions (continued)

Field	Description
16 RESERVED	This field is reserved.
17 RESERVED	This field is reserved.
18–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 START	Start DCbias and Fcap calibration in SDPLL. Once set, this bit should remain set as long as the SDPLL is being used in order to maintain calibration settings. 0 Stop DCbias and/or Fcap calibration in SDPLL 1 Start DCbias and/or Fcap calibration in SDPLL.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.6 SDPLL Control Register 3 (AFE_PLLCTRL3)

Address: 0h base + 14h offset = 14h



AFE_PLLCTRL3 field descriptions

Field	Description
0–15 RESERVED	This field is reserved.
16 RESERVED	This field is reserved.
17 RESERVED	This field is reserved.
18 LW64	SDPLL calibration wait time adjustment. 0 Default wait time of 256 reference clock cycles is used. 1 A wait time of 64 reference clock cycles is used.

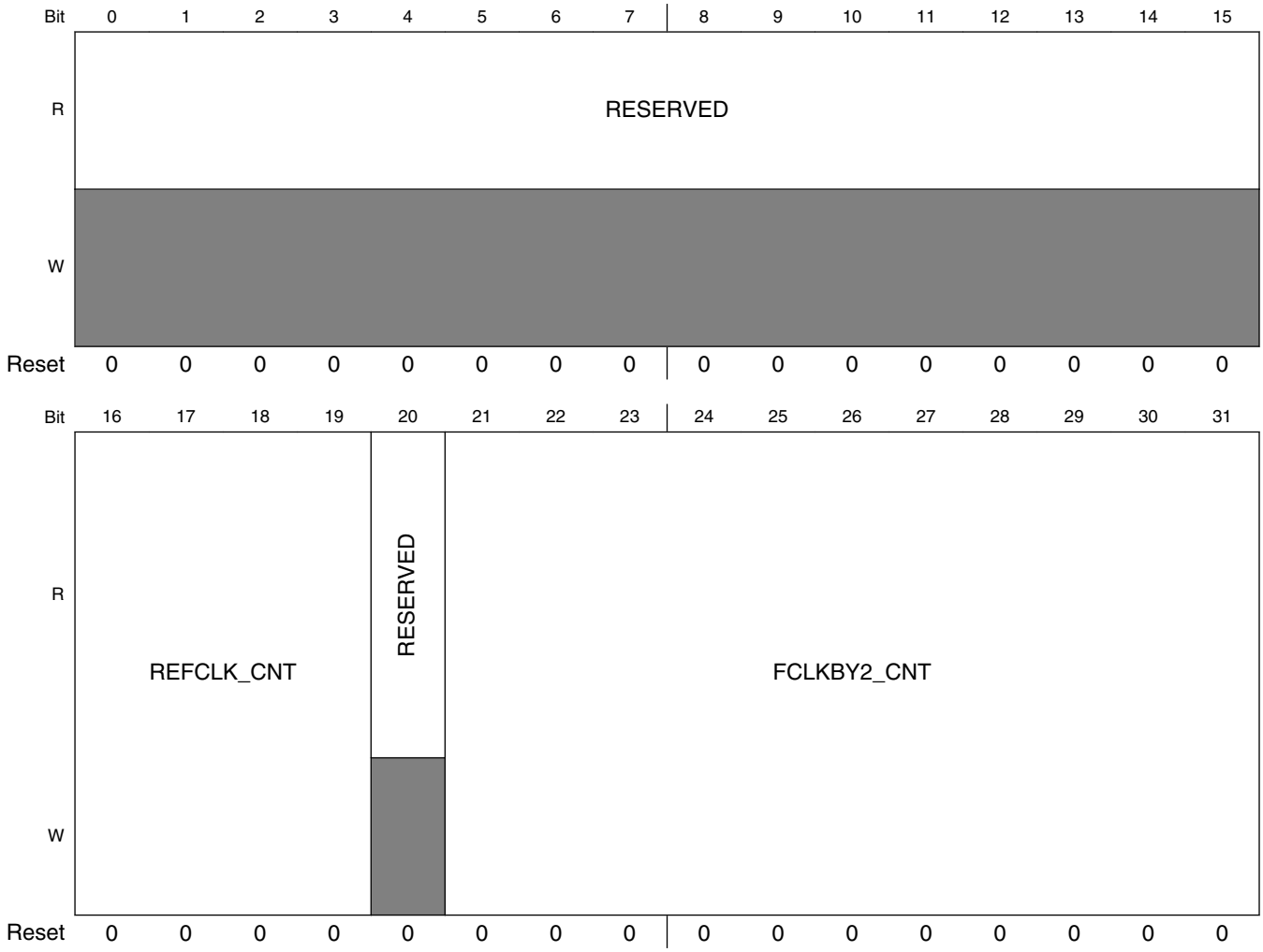
Table continues on the next page...

AFE_PLLCTRL3 field descriptions (continued)

Field	Description
19 RESERVED	This field is reserved.
20 LORIE	Loss of reference interrupt enable. This bit enables the SDPLL loss of reference interrupt to be generated when PLLSTS[LOR] is set. 0 SDPLL loss of reference interrupt disabled. 1 SDPLL loss of reference interrupt enabled.
21–23 CP_I_SEL	Charge pump current select bits. 000 001 010 011 100 Recommended value. 101 110 111
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 LCKHIIE	SDPLL lock high interrupt enable. This bit enables the SDPLL lock interrupt to be generated when the PLLSTS[LOCK] bit is high. This indicates when the SDPLL has achieved lock. 0 SDLL interrupt disabled. Interrupt will not assert when SDPLL locks. 1 SDPLL interrupt enabled. Interrupt will assert when SDPLL locks.
31 LCKLOIE	SDPLL lock loss interrupt enable. This bit enables the SDPLL lock loss interrupt to be generated when the PLLSTS[LCKLOSS] bit is high. This indicates when the SDPLL has lost lock. 0 SDPLL lock loss interrupt disabled. Interrupt will not assert when SDPLL loses lock. 1 SDPLL lock loss interrupt enabled. Interrupt will assert when SDPLL loses lock.

48.3.7 SDPLL Control Register 8 (AFE_PLLCTRL8)

Address: 0h base + 28h offset = 28h

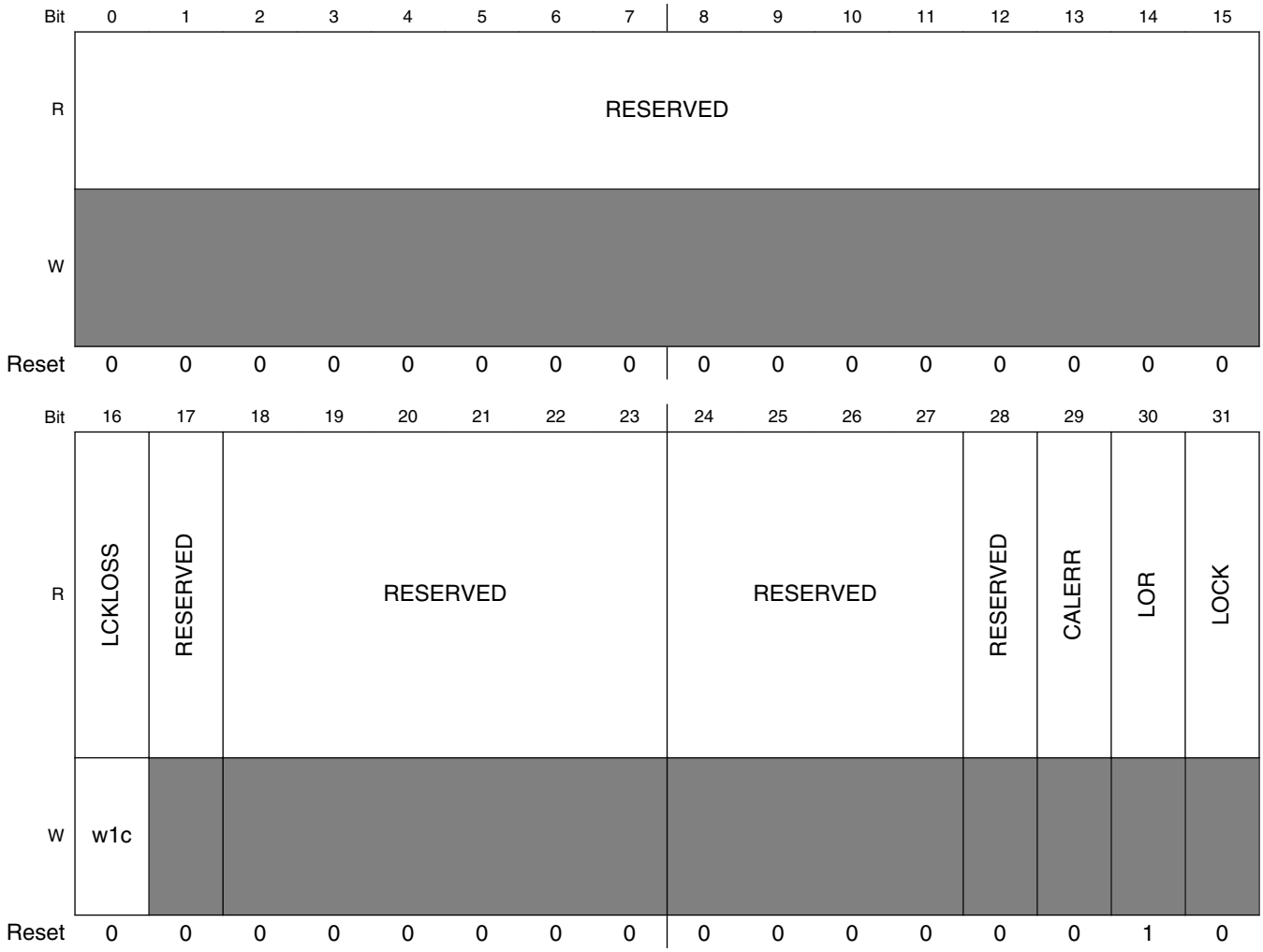


AFE_PLLCTRL8 field descriptions

Field	Description
0–15 RESERVED	This field is reserved.
16–19 REFCLK_CNT	Programmable reference clock count This field determines the number of 40MHz clock periods over which the SDPLL output is counted during calibration.
20 RESERVED	This field is reserved.
21–31 FCLKBY2_CNT	Programmable Fvco/2 clock count This field is the target count for the SDPLL VCO during calibration.

48.3.8 SDPLL Status Register (AFE_PLLSTS)

Address: 0h base + 2Ch offset = 2Ch



AFE_PLLSTS field descriptions

Field	Description
0–15 RESERVED	This field is reserved.
16 LCKLOSS	SDPLL loss of lock indicator. This bit is set when the SDPLL falls out of lock. This is determined by looking for a falling edge of PLLSTS[LOCK] while the SDPLL is powered up. When this bit is high, it can create the SDPLL loss of lock interrupt if PLLCTRL3[LCKLOIE] is set. Clear this bit by writing a 1 to this position. 0 SDPLL has not fallen out of lock. 1 SDPLL has fallen out of lock.
17 RESERVED	This field is reserved.
18–23 RESERVED	This field is reserved.

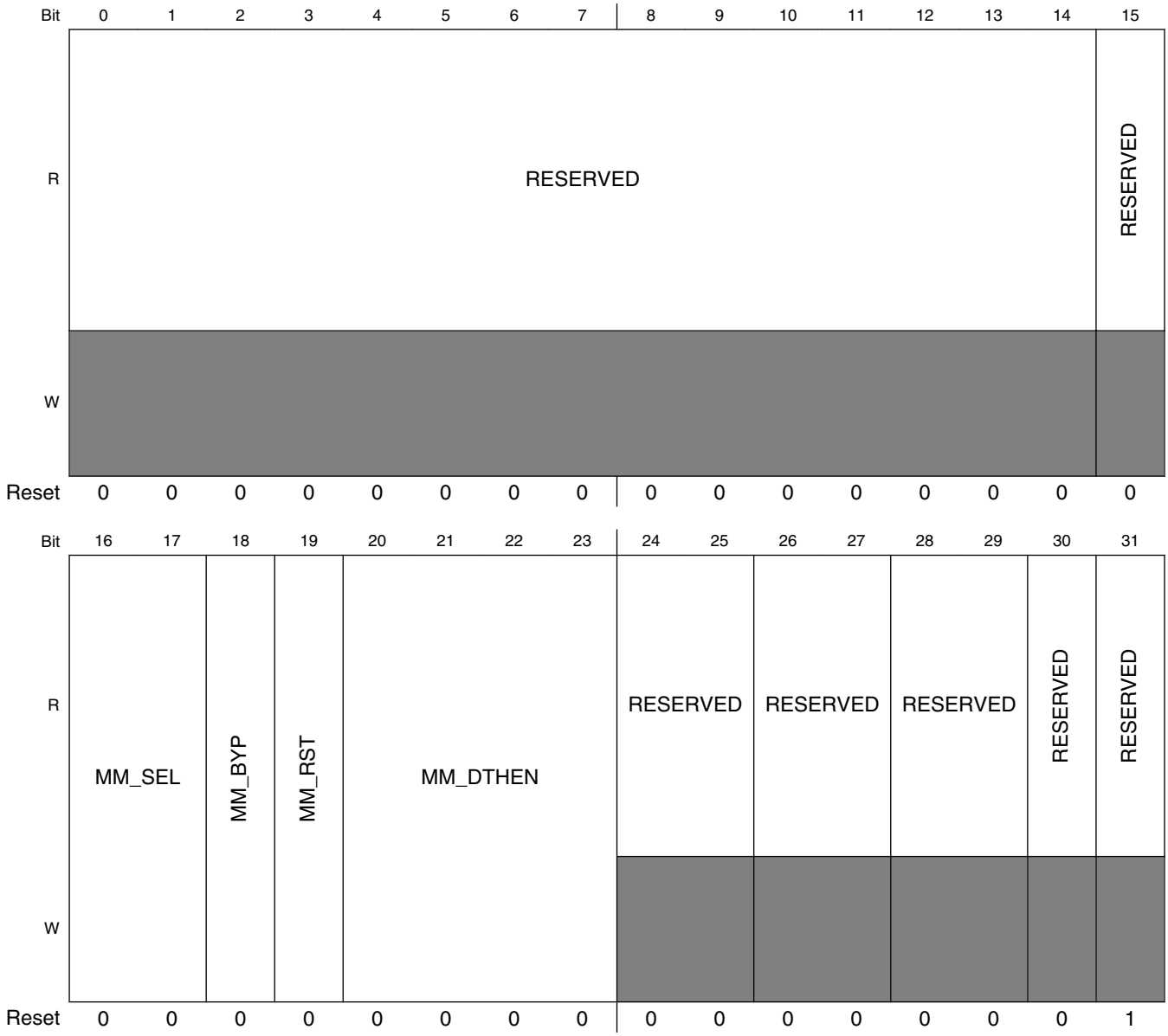
Table continues on the next page...

AFE_PLLSTS field descriptions (continued)

Field	Description
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 CALERR	SDPLL calibration error. 0 SDPLL calibration successful. 1 An error occurred during SDPLL calibration.
30 LOR	SDPLL loss of reference. This bit indicates that the SDPLL reference clock (crystal oscillator) is not toggling. This bit will be asserted about 25 clock cycles after the last edge on the reference clock. This bit will be cleared when the reference clock toggles again. This bit can be used with PLLCTRL3[LORIE] to create the SDPLL loss of reference interrupt. 0 SDPLL reference clock is toggling. 1 SDPLL reference clock is not toggling.
31 LOCK	SDPLL lock indicator This bit indicates that the SDPLL is locked. 0 SDPLL has not locked. 1 SDPLL has calibrated and locked.

48.3.9 ADC Control Register 1 (AFE_ADCCTRL1)

Address: 0h base + 30h offset = 30h



AFE_ADCCTRL1 field descriptions

Field	Description
0–14 RESERVED	This field is reserved.
15 RESERVED	This field is reserved.
16–17 MM_SEL	Mismatch shaper clock select.

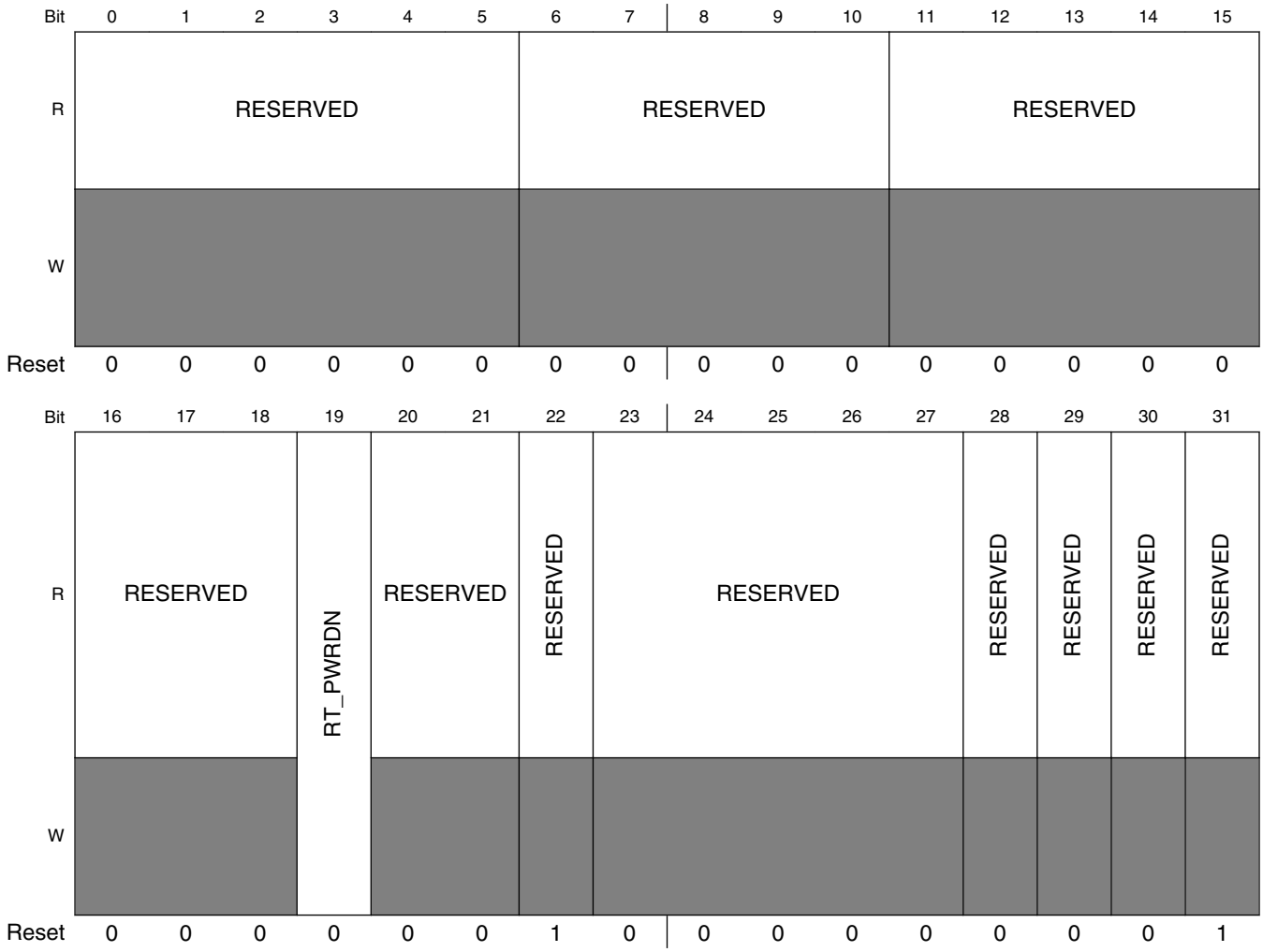
Table continues on the next page...

AFE_ADCCTRL1 field descriptions (continued)

Field	Description
	00 Bypass = 320 MHz 01 Divide by 2 = 160 MHz 10 Divide by 4 = 80 MHz 11 Unused
18 MM_BYP	Mismatch shaping algorithm bypass. 0 Mismatch shaper is enabled. 1 Mismatch shaper is bypassed.
19 MM_RST	Mismatch shaper reset. 0 Normal operation. 1 Output of mismatch shaper is taken to midscale.
20–23 MM_DTHEN	Mismatch shaper dither enable. This field is a 4-bit thermometer code to enable dither within the mismatch shaper. Each bit enables a different point in the algorithm to add dither.
24–25 RESERVED	This field is reserved.
26–27 RESERVED	This field is reserved.
28–29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.10 ADC Control Register 2 (AFE_ADCCTRL2)

Address: 0h base + 34h offset = 34h



AFE_ADCCTRL2 field descriptions

Field	Description
0–5 RESERVED	This field is reserved.
6–10 RESERVED	This field is reserved.
11–15 RESERVED	This field is reserved.
16–18 RESERVED	This field is reserved.
19 RT_PWRDN	External resistor trim circuit powerdown control bit. **NOTE:** Do not set this bit if there is no external resistor connected to the SD_R pin.

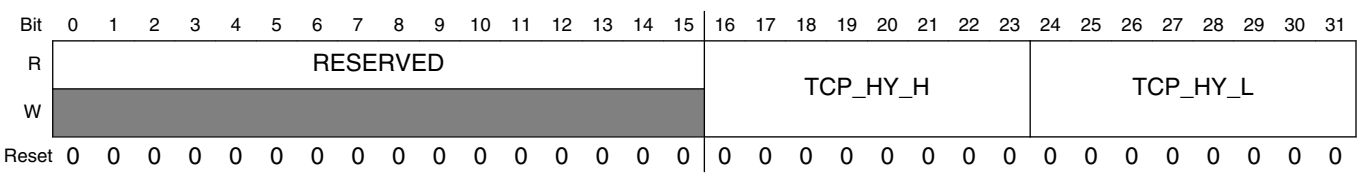
Table continues on the next page...

AFE_ADCCTRL2 field descriptions (continued)

Field	Description
	0 Enable external resistor trim. 1 Disable, powerdown. Forces ADCTRIM[RESRDY] low.
20–21 RESERVED	This field is reserved.
22 RESERVED	This field is reserved.
23–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.11 ADC Tracking Oscillator Trim Register (AFE_ADCTOT)

Address: 0h base + 38h offset = 38h

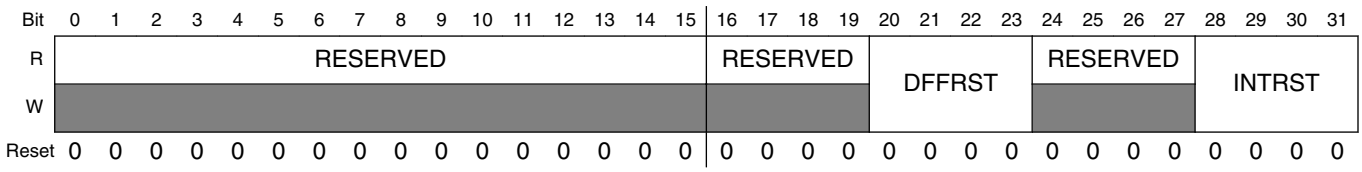


AFE_ADCTOT field descriptions

Field	Description
0–15 RESERVED	This field is reserved.
16–23 TCP_HY_H	Provides trim capability to improve accuracy of the tracking oscillator.
24–31 TCP_HY_L	Provides trim capability to improve accuracy of the tracking oscillator.

48.3.12 ADC Reset Register (AFE_ADCRST)

Address: 0h base + 3Ch offset = 3Ch

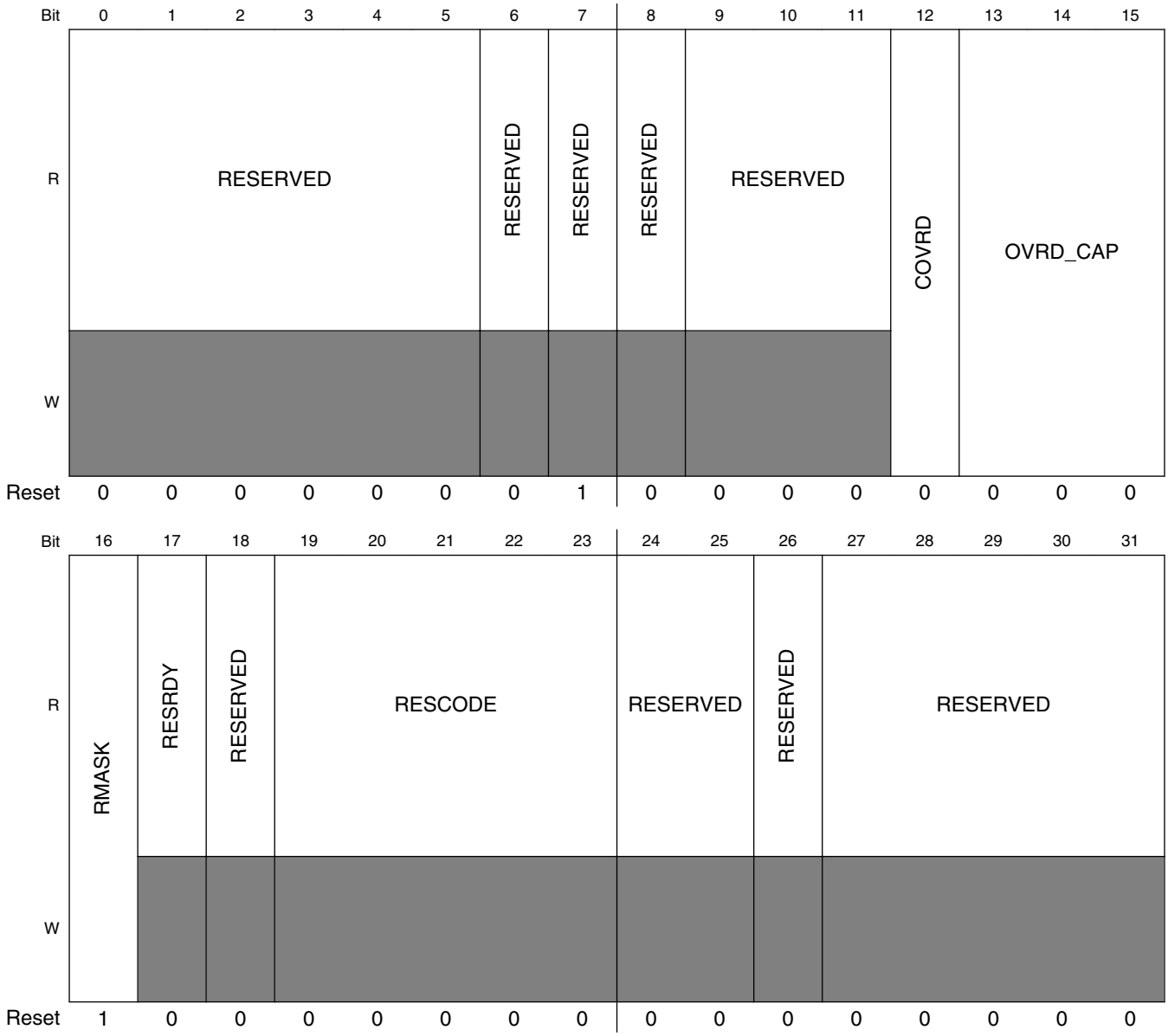


AFE_ADCRST field descriptions

Field	Description
0–15 RESERVED	This field is reserved.
16–19 RESERVED	This field is reserved.
20–23 DFFRST	Reset control for the flip flops for the digital word to the DAC's in the eight ADC's. 0 Normal operation. 1 Reset code word of a specific ADC to midscale of the DAC's.
24–27 RESERVED	This field is reserved.
28–31 INTRST	Zeros integrator capacitor charge of each of the four ADC's. 0 Normal operation. 1 Sets the output of the integrator of a specific ADC at analog ground.

48.3.13 ADC Trim Register (AFE_ADCTRIM)

Address: 0h base + 40h offset = 40h



AFE_ADCTRIM field descriptions

Field	Description
0–5 RESERVED	This field is reserved.
6 RESERVED	This field is reserved.
7 RESERVED	This field is reserved.

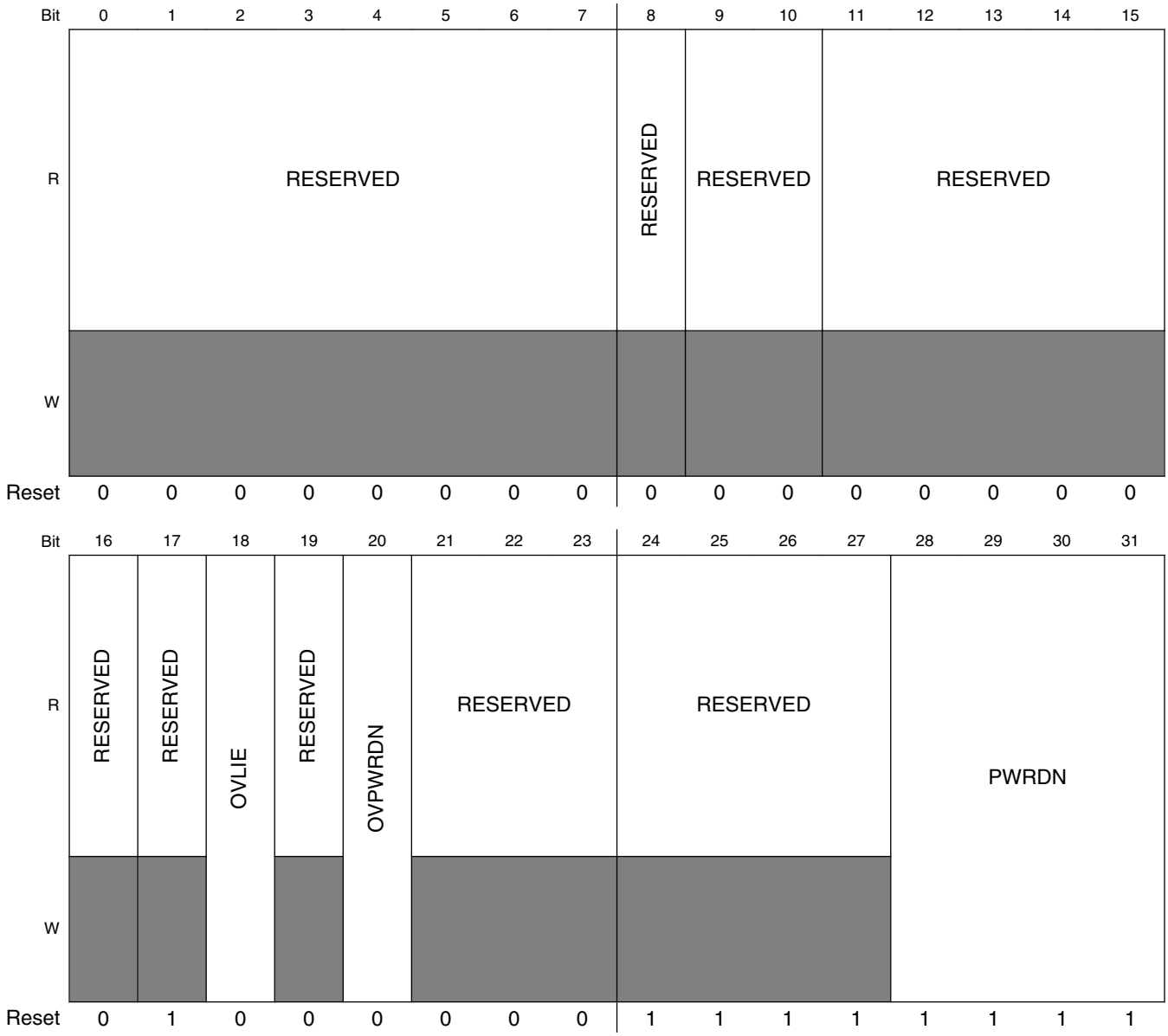
Table continues on the next page...

AFE_ADCTRIM field descriptions (continued)

Field	Description
8 RESERVED	This field is reserved.
9–11 RESERVED	This field is reserved.
12 COVRD	Capacitor Trim override control bit. 0 Normal operation. 1 Override Capacitor Trim value with ADCTRIM[OVRD_CAP].
13–15 OVRD_CAP	Capacitor trim override value. This value is only used when ADCTRIM[COVRD] is set. It is then driven into the capacitor trim circuitry for use in place of the value computed from the trimming.
16 RMASK	Resistor arrays trim mask. To start the ADC trim procedure set this bit to 1 and then clear it. Trimming can only be performed when ADCCTRL2[RT_PWRDN] is 0. 0 Enables the resistor arrays to be trimmed. 1 The resistor array is set to default midcode setting.
17 RESRDY	External resistor trim flag. This bit is the output from the restrim comparator to indicate when the correct resistor trim value has been reached. 0 Resistor trim routine is not complete. 1 Resistor trim routine is complete.
18 RESERVED	This field is reserved.
19–23 RESCODE	The correct trim value for the resistor arrays within the Vref and the resistor trim.
24–25 RESERVED	This field is reserved.
26 RESERVED	This field is reserved.
27–31 RESERVED	This field is reserved.

48.3.14 ADC Control Register 7 (AFE_ADCCTRL7)

Address: 0h base + 48h offset = 48h



AFE_ADCCTRL7 field descriptions

Field	Description
0–7 RESERVED	This field is reserved.
8 RESERVED	This field is reserved.
9–10 RESERVED	This field is reserved.

Table continues on the next page...

AFE_ADCCTRL7 field descriptions (continued)

Field	Description
11–15 RESERVED	This field is reserved.
16 RESERVED	This field is reserved.
17 RESERVED	This field is reserved.
18 OVLIE	ADC overvoltage detect interrupt enable. 0 Disable overvoltage detect interrupt. 1 Create overvoltage detect interrupt if any of ADCOVLD[OVERVOLTS] is set.
19 RESERVED	This field is reserved.
20 OVPWRDN	ADC overvoltage protection circuitry powerdown. 0 Overvoltage protection on (channels 0-3). 1 Overvoltage protection circuitry off and powered down (channels 0-3).
21–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28–31 PWRDN	ADC powerdown enables for ADC 3 down to 0 (also referred to as ADC D down to A). 0 Enable one of the four ADC's. 1 Disable, powerdown one of the four ADC's. **NOTE:** It is recommended to not power up all ADCs simultaneously.

48.3.15 ADC Overload Detect Register (AFE_ADCOVLD)

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED				OVERVOLT				RESERVED				OVERVOLTS				RESERVED															
W													w1c																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AFE_ADCOVLD field descriptions

Field	Description
0–3 RESERVED	This field is reserved.
4–7 OVERVOLT	ADC overvoltage detect flag for each of the four ADC's. These bits reflect the realtime status of the ADC overvoltage detect circuits. An overvoltage condition (ADC absolute input voltage greater than 1.2V) will be acted upon immediately by the analog circuitry but must last at least 12.5nsec to cause a flag to set. A flag will clear once the overvoltage condition has gone away for at least 12.5nsec.

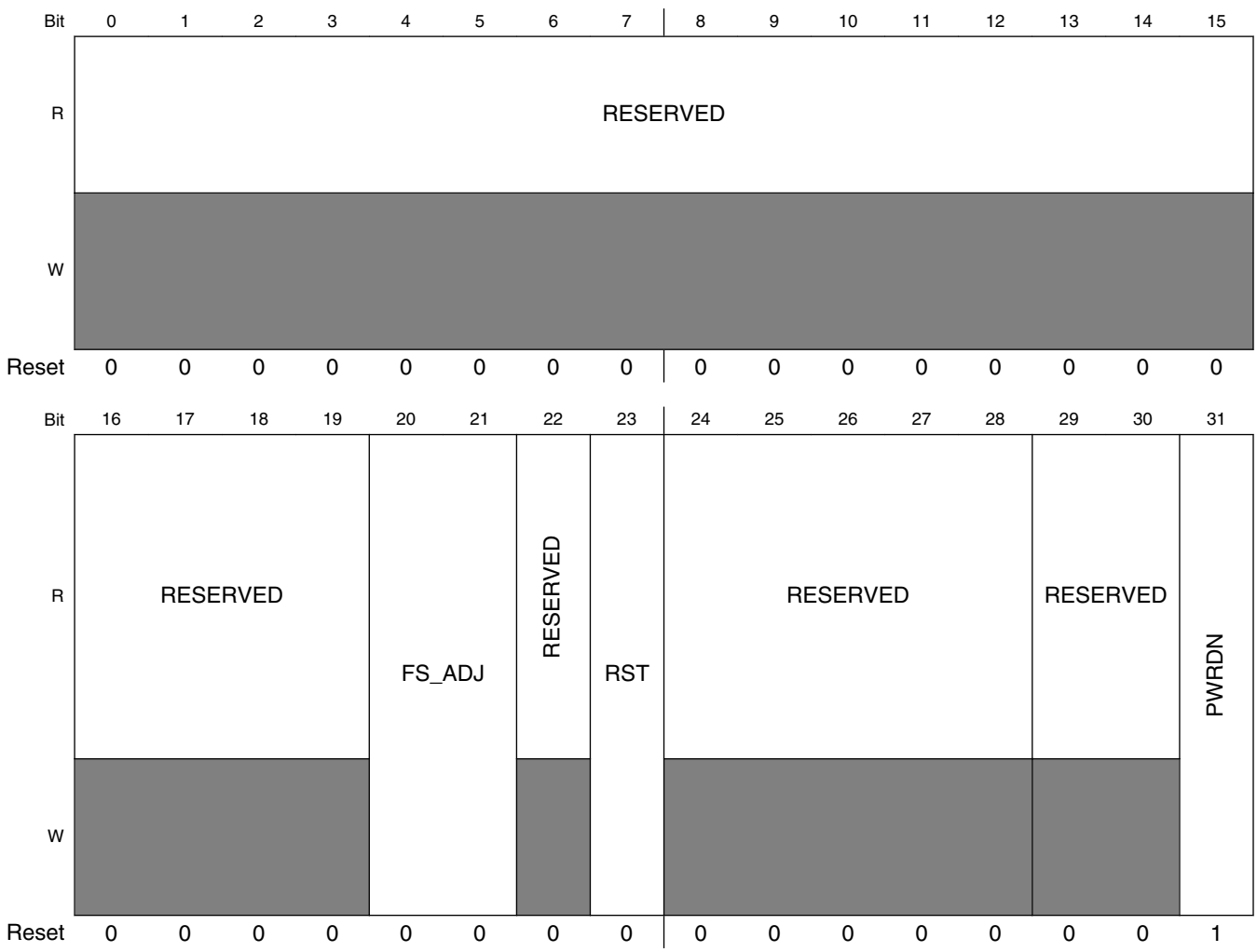
Table continues on the next page...

AFE_ADCOVLD field descriptions (continued)

Field	Description
8–11 RESERVED	This field is reserved.
12–15 OVERVOLTS	ADC overvoltage detect flag for each of the four ADC's. These bits are sticky and will remain asserted once set. Clear each bit by writing a 1 to that bit position.
16–31 RESERVED	This field is reserved.

48.3.16 DAC Control Register (AFE_DACCTRL)

Address: 0h base + 50h offset = 50h

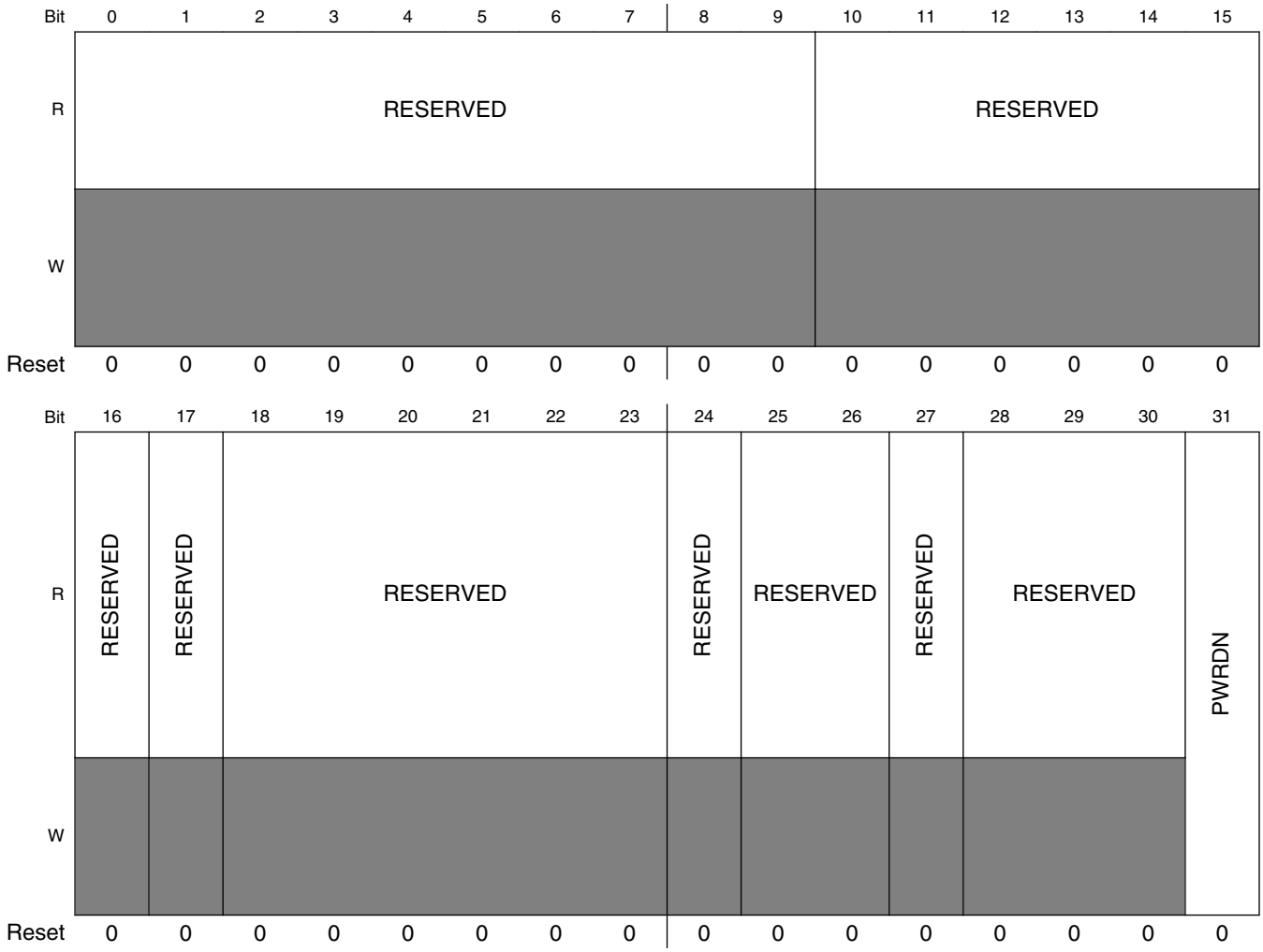


AFE_DACCTRL field descriptions

Field	Description
0–19 RESERVED	This field is reserved.
20–21 FS_ADJ	DAC full scale adjust. Increases the full scale current to overcome the DAC region overlaps. 00 Additional 240µA will flow to dacoutm. 01 Additional 120µA will flow to dacoutp and to dacoutm. 10 Additional 120µA will flow to dacoutp and to dacoutm. 11 Additional 240µA will flow to dacoutp.
22 RESERVED	This field is reserved.
23 RST	DAC reset. Resets the DAC outputs to 1.272V and 0.04V when the DAC is clocked. 0 Normal operation. 1 Resets DAC to midscale.
24–28 RESERVED	This field is reserved.
29–30 RESERVED	This field is reserved.
31 PWRDN	DAC powerdown. 0 Enable DAC. 1 Disable, powerdown.

48.3.17 VREF Control Register 1 (AFE_VRFCTRL1)

Address: 0h base + 54h offset = 54h



AFE_VRFCTRL1 field descriptions

Field	Description
0–9 RESERVED	This field is reserved.
10–15 RESERVED	This field is reserved.
16 RESERVED	This field is reserved.
17 RESERVED	This field is reserved.
18–23 RESERVED	This field is reserved.

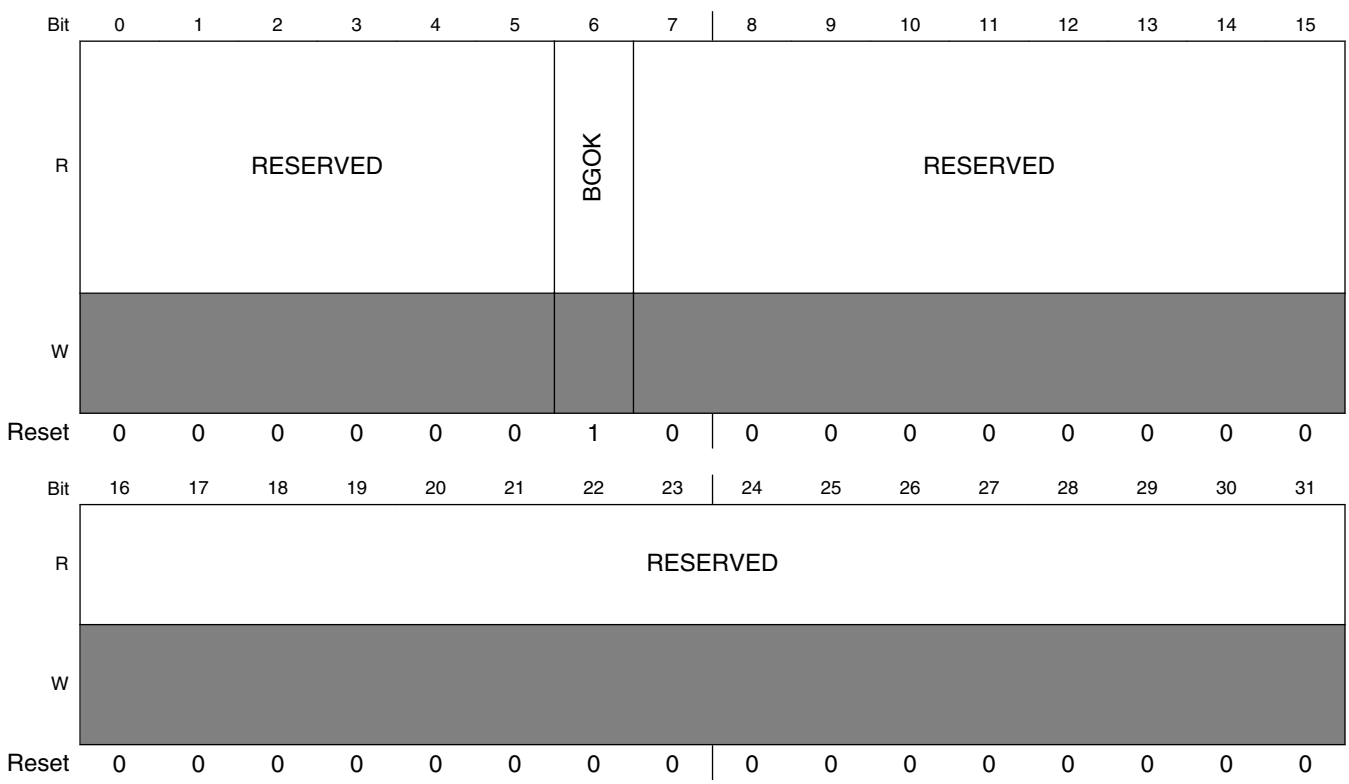
Table continues on the next page...

AFE_VRFCTRL1 field descriptions (continued)

Field	Description
24 RESERVED	This field is reserved.
25–26 RESERVED	This field is reserved.
27 RESERVED	This field is reserved.
28–30 RESERVED	This field is reserved.
31 PWRDN	Vref powerdown. 0 Enable. 1 Disable, powerdown. Will not powerdown the bandgap, the VREG LVD's, or the current sources for the OSC and SDPLL.

48.3.18 Low Voltage Detect Status Register (AFE_LVDSTS)

Address: 0h base + 5Ch offset = 5Ch



AFE_LVDSTS field descriptions

Field	Description
0–5 RESERVED	This field is reserved.
6 BGOK	1.2V Vref is up and available. 0 1.2V Vref is not present. 1 1.2V Vref is present.
7–31 RESERVED	This field is reserved.

48.3.19 VREG Reserved Register (AFE_VRGRSVD)

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

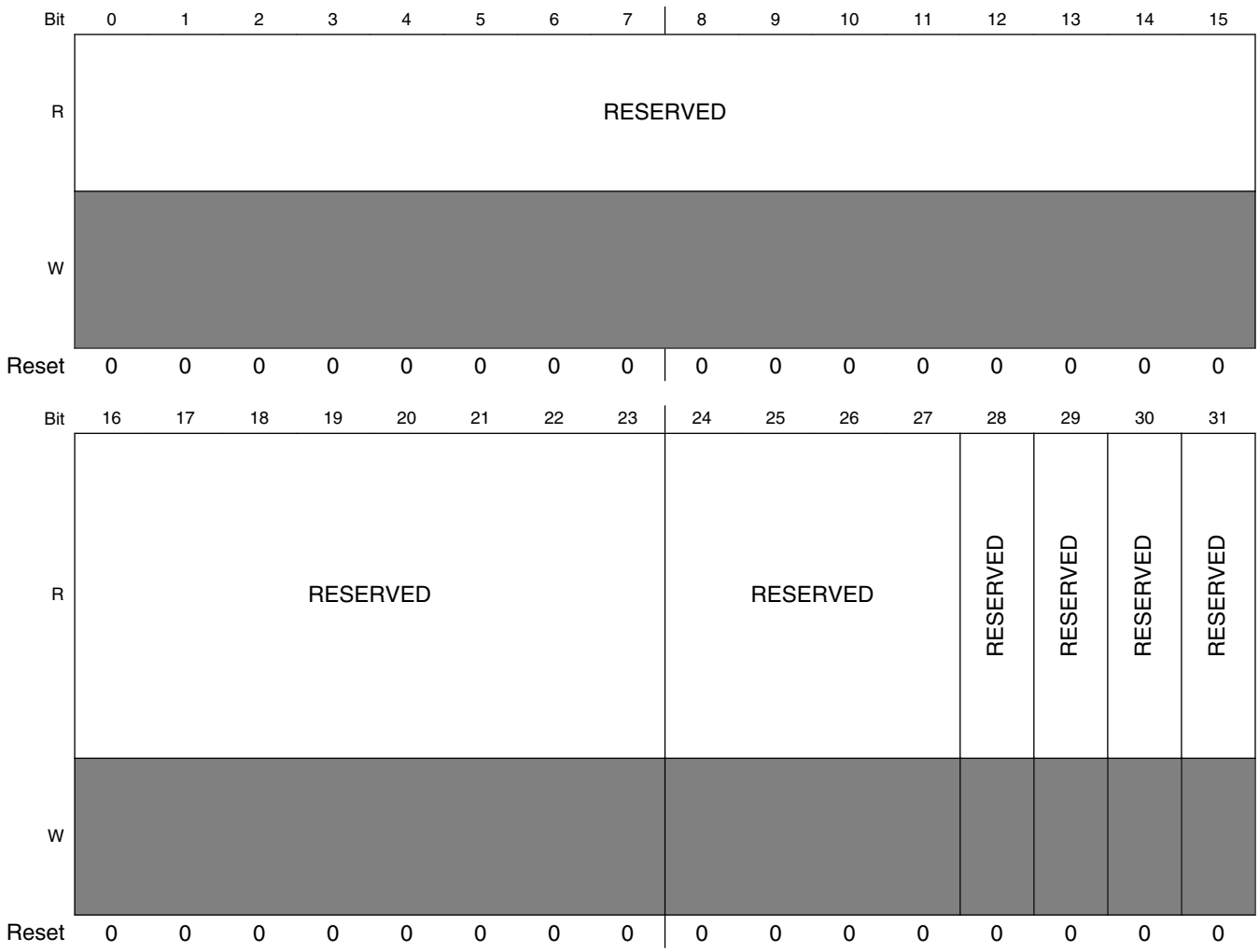
AFE_VRGRSVD field descriptions

Field	Description
0–31 RESERVED	This field is reserved.

48.3.20 VREG2 Control Register (AFE_VRGCTRL2)

This register controls the operation of the voltage regulator for the AFE ADC analog circuitry.

Address: 0h base + 64h offset = 64h



AFE_VRGCTRL2 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.

Table continues on the next page...

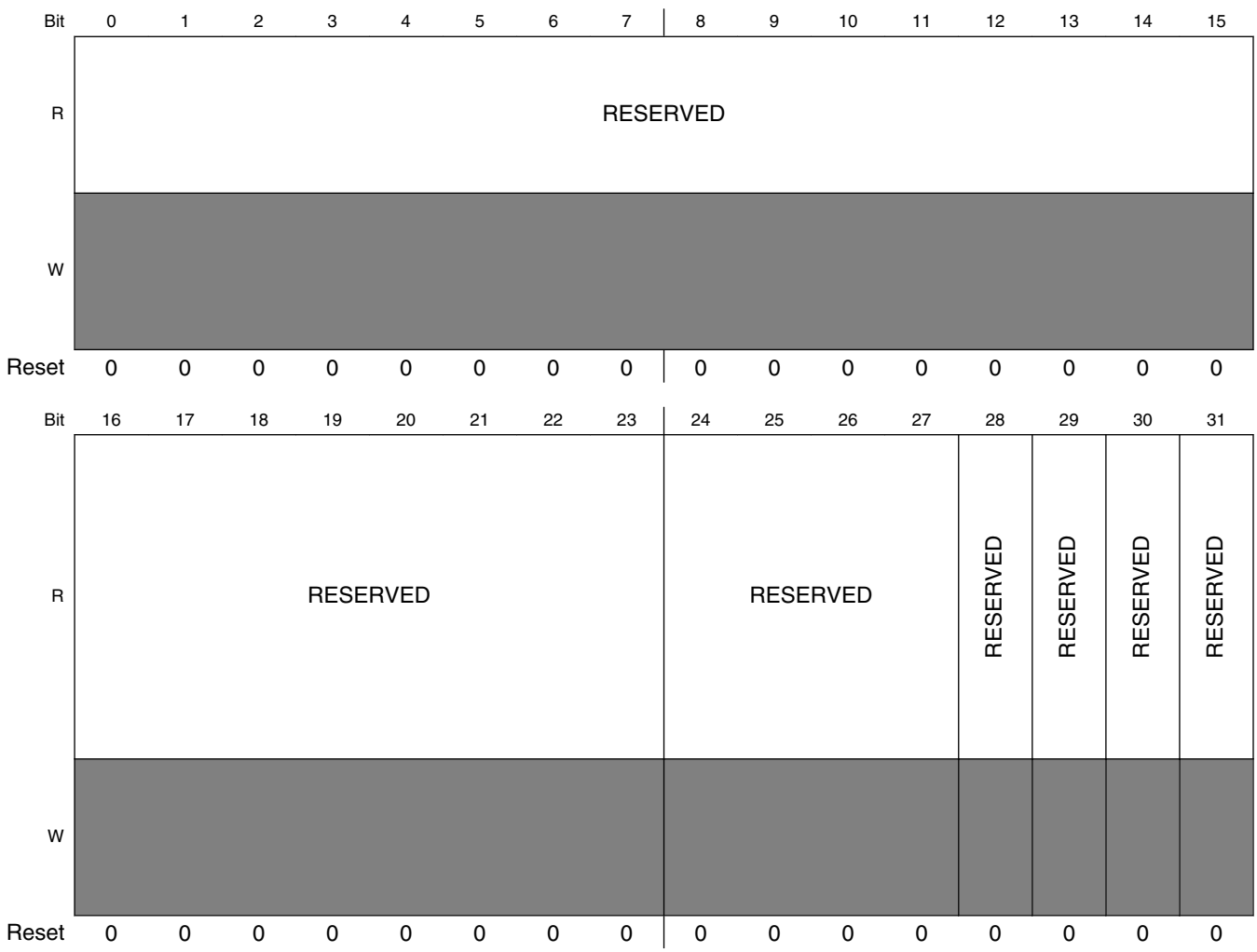
AFE_VRGCTRL2 field descriptions (continued)

Field	Description
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.21 VREG3 Control Register (AFE_VRGCTRL3)

This register controls the operation of the voltage regulator for the AFE ADC digital circuitry.

Address: 0h base + 68h offset = 68h



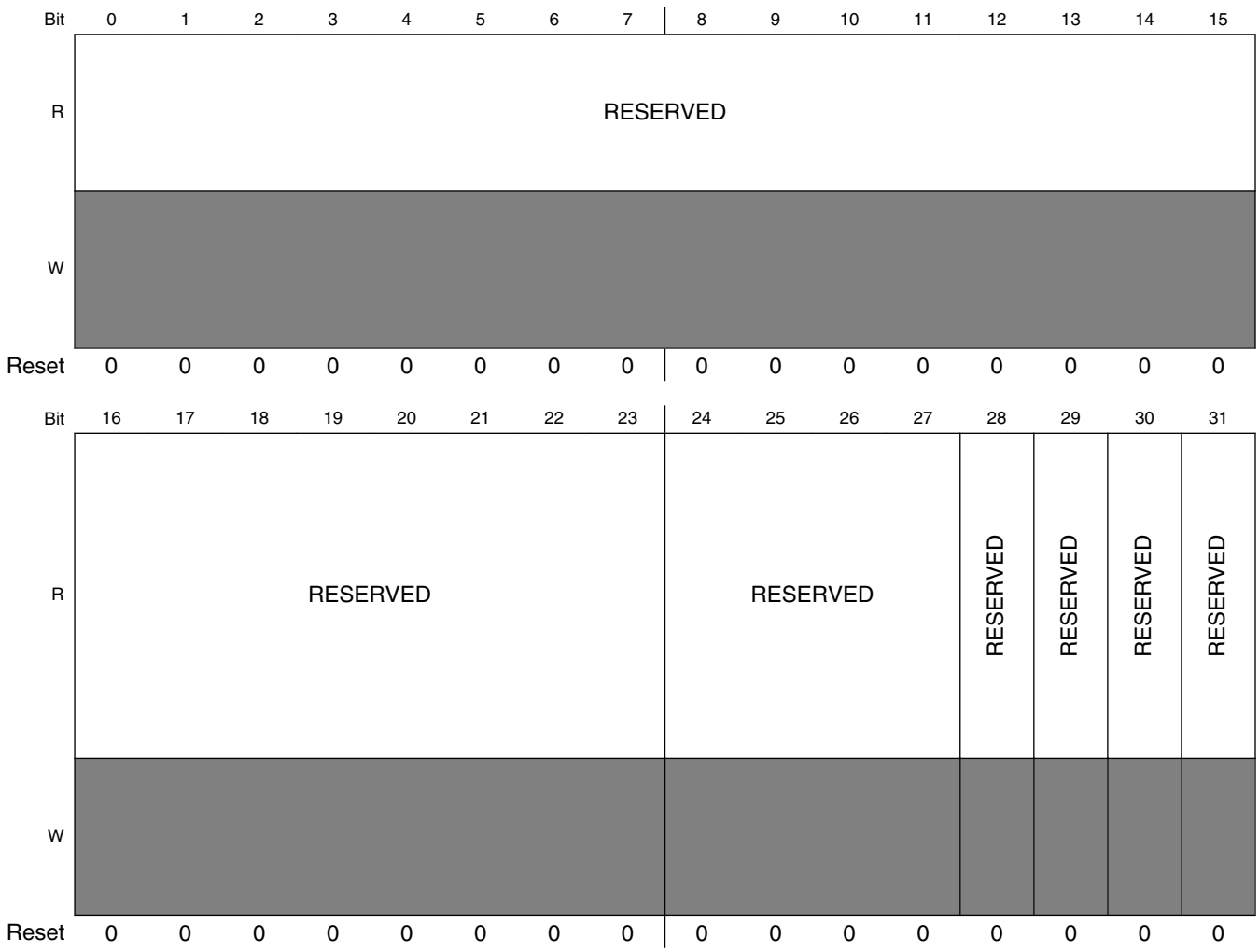
AFE_VRGCTRL3 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.22 VREG4 Control Register (AFE_VRGCTRL4)

This register controls the operation of the voltage regulator for the AFE DAC external capacitor circuitry.

Address: 0h base + 6Ch offset = 6Ch



AFE_VRGCTRL4 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.

Table continues on the next page...

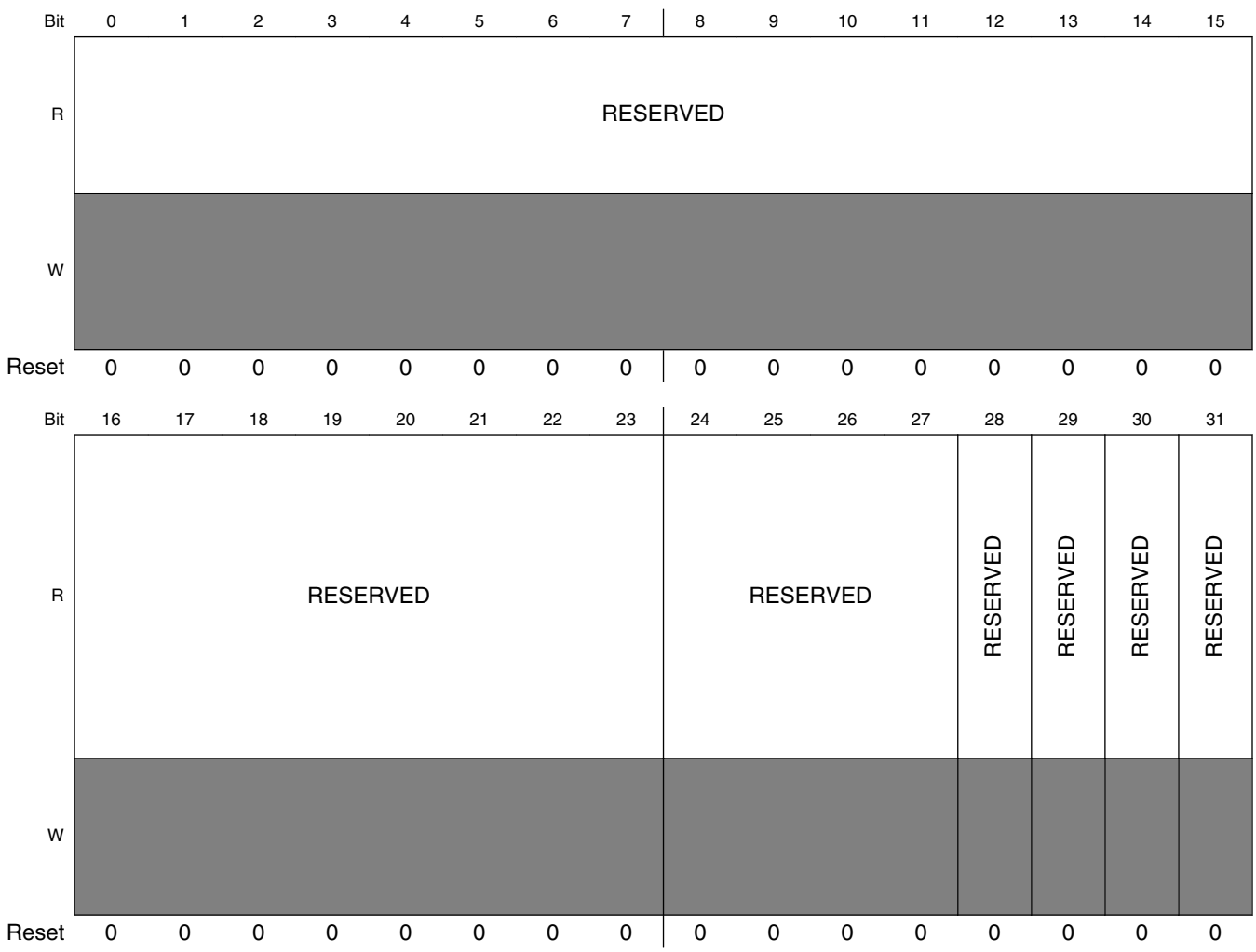
AFE_VRGCTRL4 field descriptions (continued)

Field	Description
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.23 VREG5 Control Register (AFE_VRGCTRL5)

This register controls the operation of the voltage regulator for the AFE OSC circuitry.

Address: 0h base + 70h offset = 70h



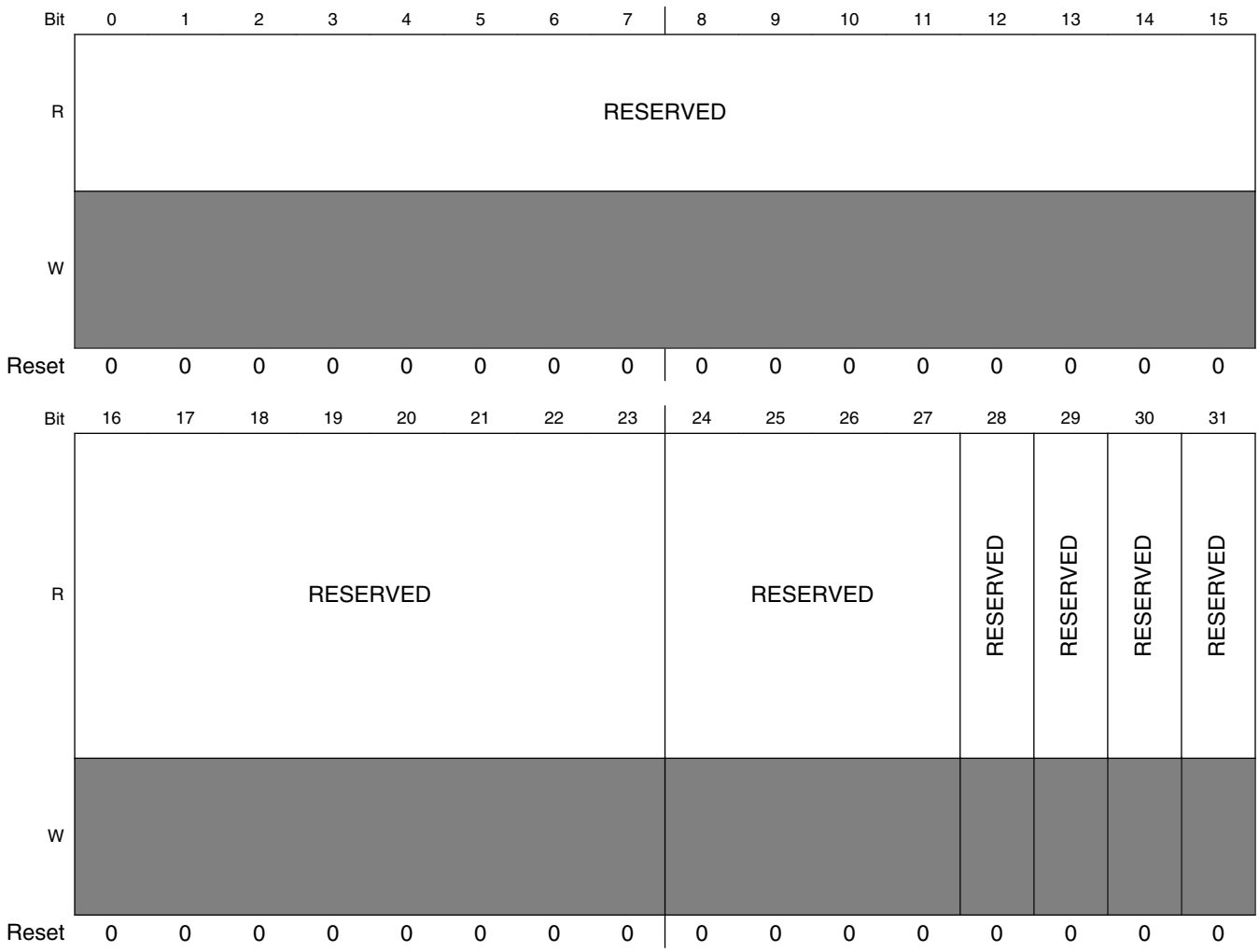
AFE_VRGCTRL5 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.24 VREG6 Control Register (AFE_VRGCTRL6)

This register controls the operation of the voltage regulator for the AFE SDPLL analog circuitry.

Address: 0h base + 74h offset = 74h



AFE_VRGCTRL6 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.

Table continues on the next page...

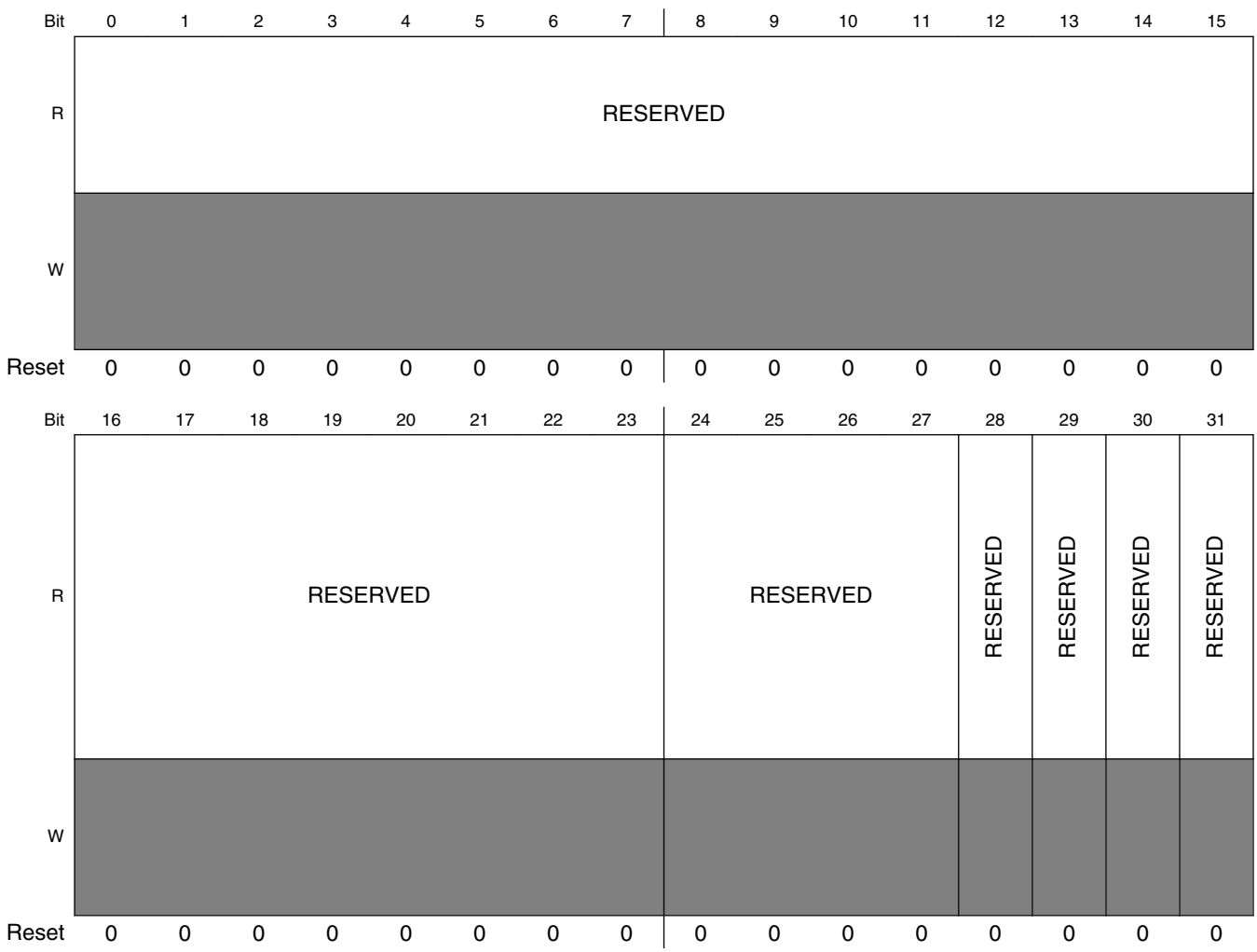
AFE_VRGCTRL6 field descriptions (continued)

Field	Description
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.25 VREG7 Control Register (AFE_VRGCTRL7)

This register controls the operation of the voltage regulator for the AFE SDPLL digital circuitry.

Address: 0h base + 78h offset = 78h



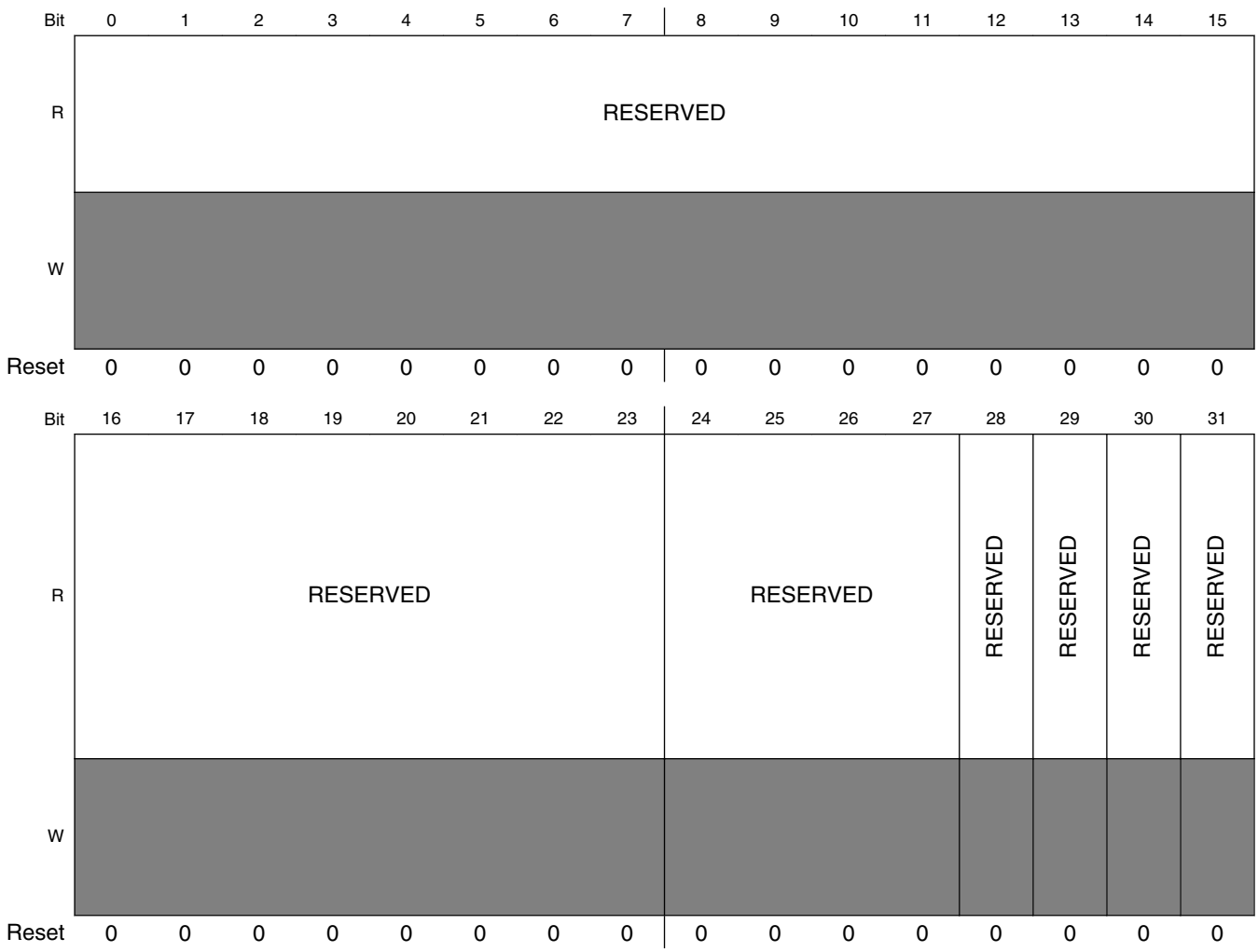
AFE_VRGCTRL7 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.26 VREG8 Control Register (AFE_VRGCTRL8)

This register controls the voltage of the level shifter logic.

Address: 0h base + 7Ch offset = 7Ch



AFE_VRGCTRL8 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.

Table continues on the next page...

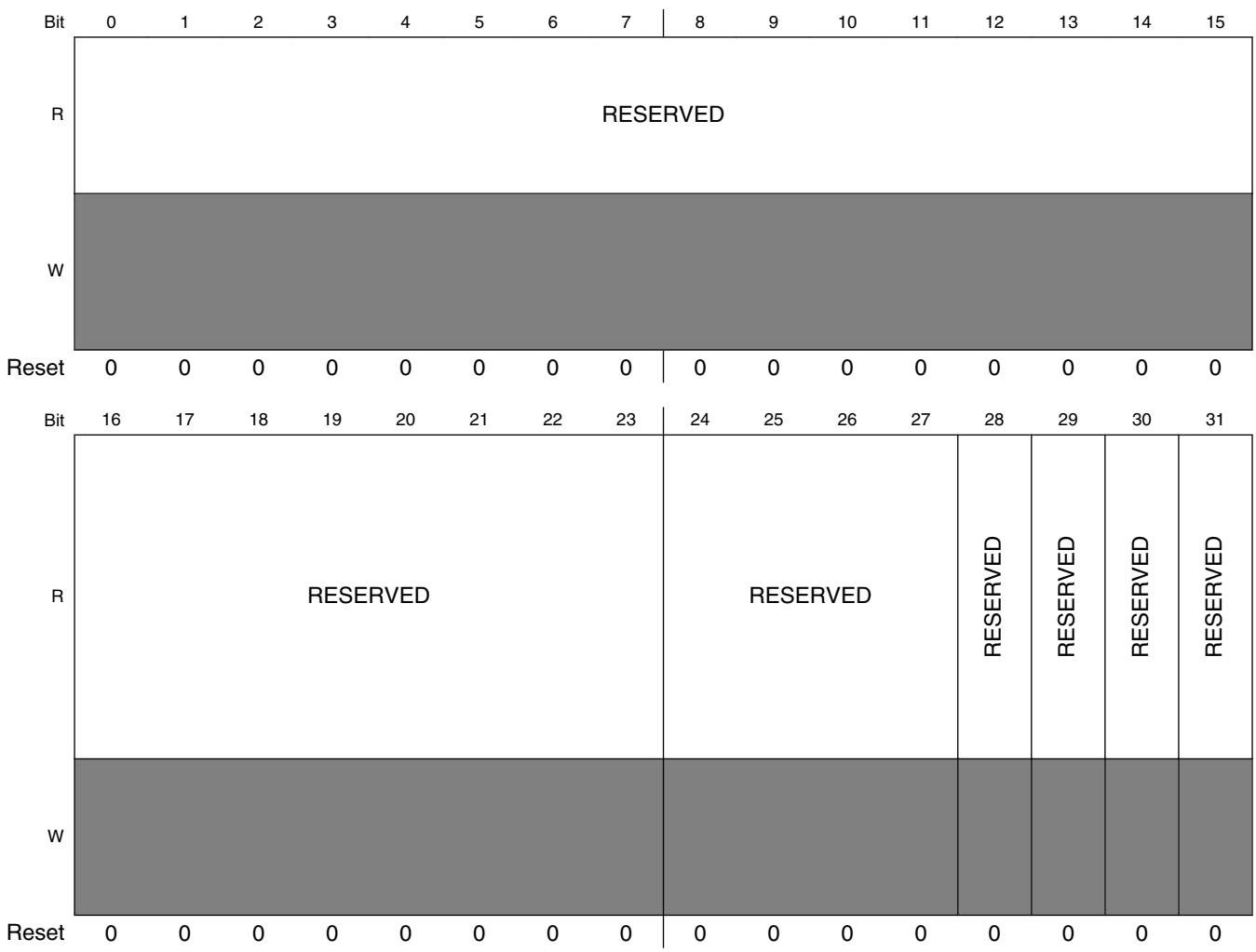
AFE_VRGCTRL8 field descriptions (continued)

Field	Description
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.27 VREG9 Control Register (AFE_VRGCTRL9)

This register controls the operation of the voltage regulator for the AFE DAC internal capacitor circuitry.

Address: 0h base + 84h offset = 84h



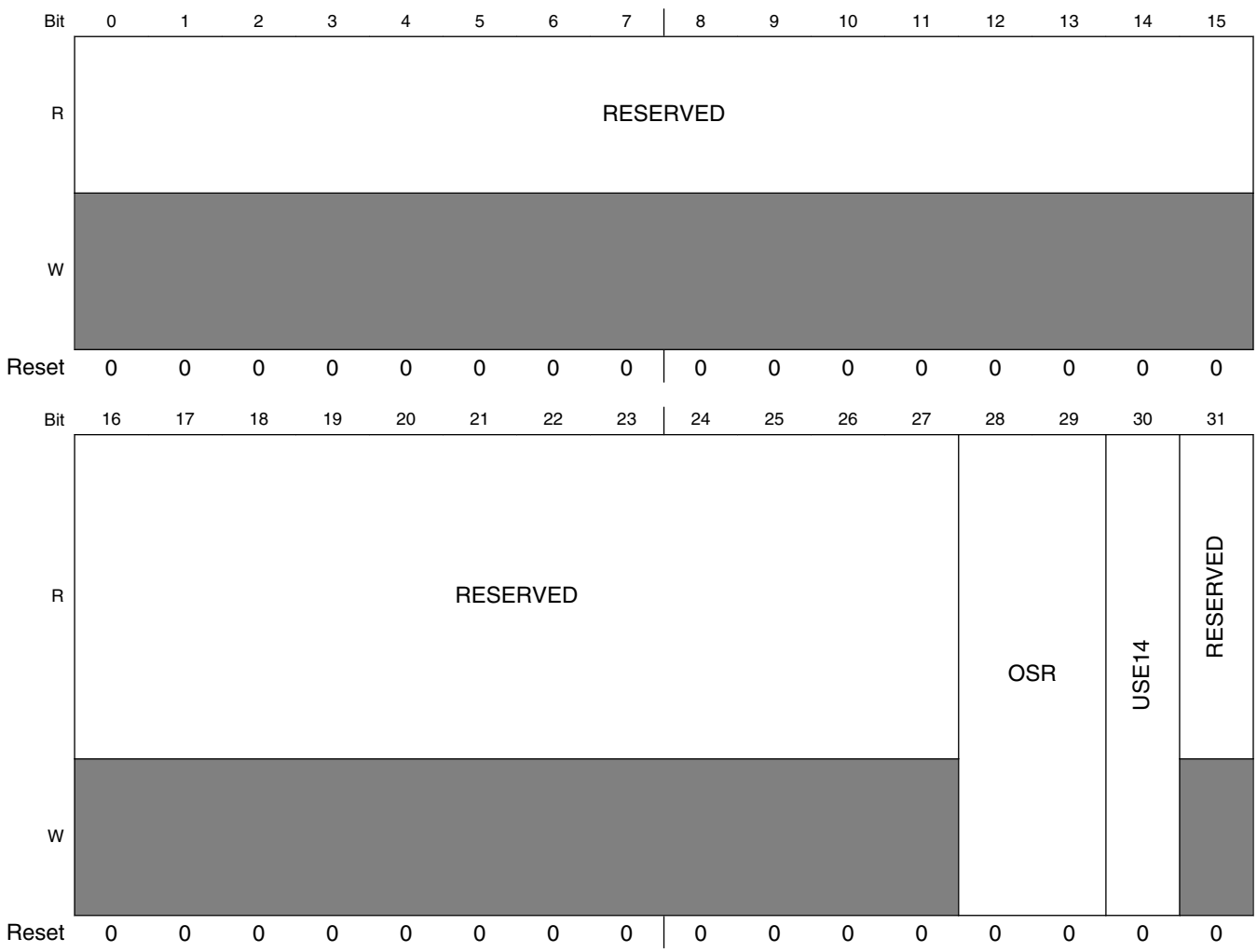
AFE_VRGCTRL9 field descriptions

Field	Description
0–23 RESERVED	This field is reserved.
24–27 RESERVED	This field is reserved.
28 RESERVED	This field is reserved.
29 RESERVED	This field is reserved.
30 RESERVED	This field is reserved.
31 RESERVED	This field is reserved.

48.3.28 Decimation Filter Control Register (AFE_FILTCTRL)

This register controls the operation of decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active.

Address: 0h base + A0h offset + (4d × i), where i=0d to 3d



AFE_FILTCTRL*n* field descriptions

Field	Description
0–27 RESERVED	This field is reserved.
28–29 OSR	This field controls the oversampling ratio of filter n (where n = 0 to 3). 00 32 times oversampling. 01 64 times oversampling.

Table continues on the next page...

AFE_FILTCTRL*n* field descriptions (continued)

Field	Description
	10 128 times oversampling. 11 256 times oversampling.
30 USE14	This bit controls the use of the full 14-bit data outputs for filter n 0 Output data is rounded to 12 bits with the 2 LSB's being forced to 00. 1 Output data is full 14 bits.
31 RESERVED	This field is reserved.

48.3.29 Decimation Filter Coefficient Register 0 (AFE_FLCOEF0)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + C0h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																FnCOEF0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	0	1

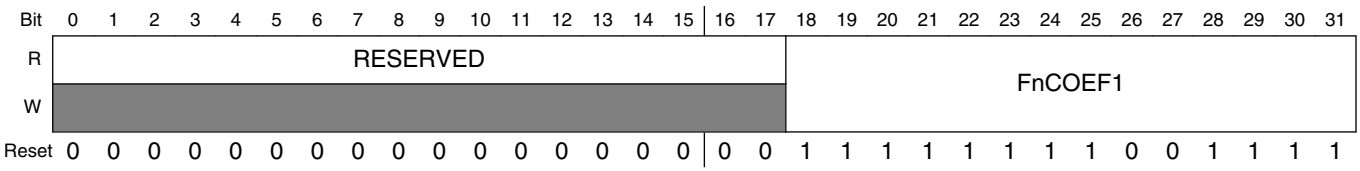
AFE_FLCOEF0*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF0	This field holds the value for coefficient 0 of decimation filter n (where n = 0 to 3).

48.3.30 Decimation Filter Coefficient Register 1 (AFE_FLCOEF1)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + C4h offset + (64d × i), where i=0d to 3d



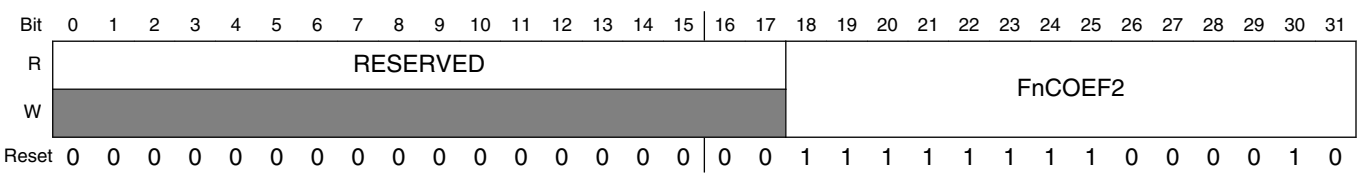
AFE_FLCOEF1*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF1	Decimation Filter n Coefficient 1. This field holds the value for coefficient 1 of decimation filter n (where n = 0 to 3).

48.3.31 Decimation Filter Coefficient Register 2 (AFE_FLCOEF2)

This holds one of the coefficients for decimation filter n. This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + C8h offset + (64d × i), where i=0d to 3d



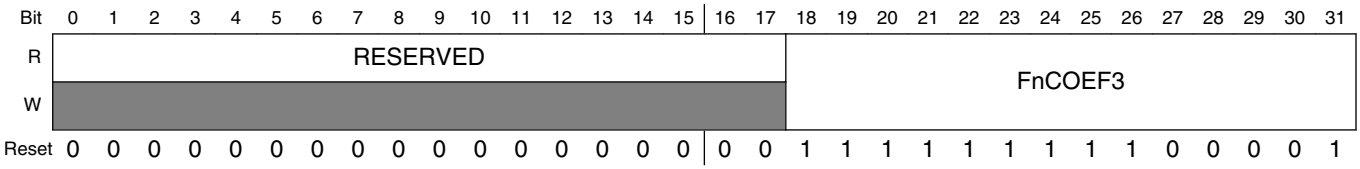
AFE_FLCOEF2*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF2	This field holds the value for coefficient 2 of decimation filter n (where n = 0 to 3).

48.3.32 Decimation Filter Coefficient Register 3 (AFE_FLCOEF3)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written by using 16-bit or 32-bit accesses.

Address: 0h base + CCh offset + (64d × i), where i=0d to 3d



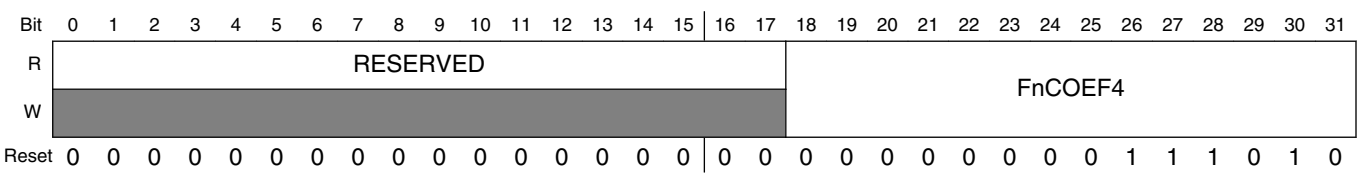
AFE_FLCOEF3*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF3	This field holds the value for coefficient 3 of decimation filter n (where n = 0 to 3).

48.3.33 Decimation Filter Coefficient Register 4 (AFE_FLCOEF4)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + D0h offset + (64d × i), where i=0d to 3d



AFE_FLCOEF4*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF4	This field holds the value for coefficient 4 of decimation filter n (where n = 0 to 3).

48.3.34 Decimation Filter Coefficient Register 5 (AFE_FLCOEF5)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + D4h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																FnCOEF5															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0

AFE_FLCOEF5*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF5	This field holds the value for coefficient 5 of decimation filter n (where n = 0 to 3).

48.3.35 Decimation Filter Coefficient Register 6 (AFE_FLCOEF6)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + D8h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																FnCOEF6															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0

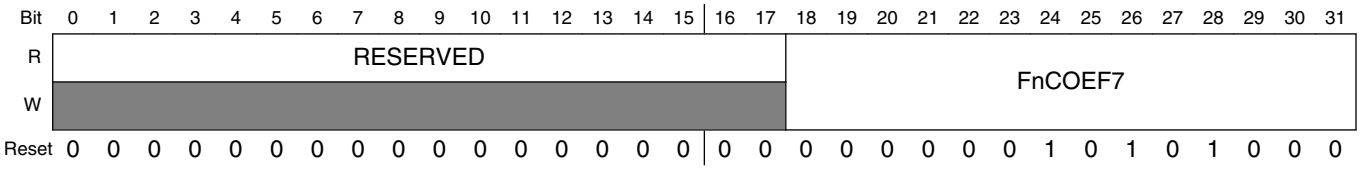
AFE_FLCOEF6*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF6	Decimation Filter n Coefficient 6. This field holds the value for coefficient 6 of decimation filter n.

48.3.36 Decimation Filter Coefficient Register 7 (AFE_FLCOEF7)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + DCh offset + (64d × i), where i=0d to 3d



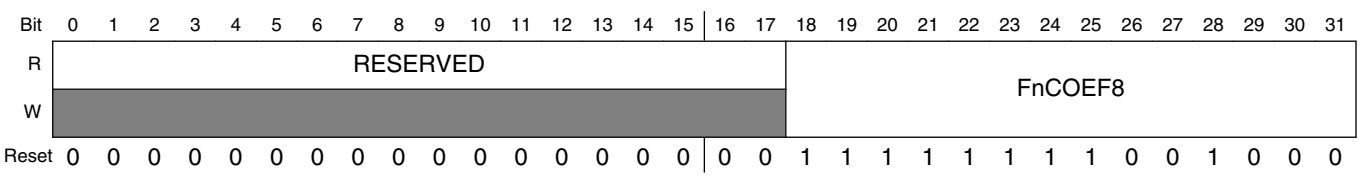
AFE_FLCOEF7*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF7	This field holds the value for coefficient 7 of decimation filter n (where n = 0 to 3).

48.3.37 Decimation Filter Coefficient Register 8 (AFE_FLCOEF8)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + E0h offset + (64d × i), where i=0d to 3d



AFE_FLCOEF8*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF8	This field holds the value for coefficient 8 of decimation filter n (where n = 0 to 3).

48.3.38 Decimation Filter Coefficient Register 9 (AFE_FLCOEF9)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + E4h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																FnCOEF9															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0

AFE_FLCOEF9*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF9	This field holds the value for coefficient 9 of decimation filter n (where n = 0 to 3).

48.3.39 Decimation Filter Coefficient Register 10 (AFE_FLCOEF10)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + E8h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESERVED																FnCOEF10															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0

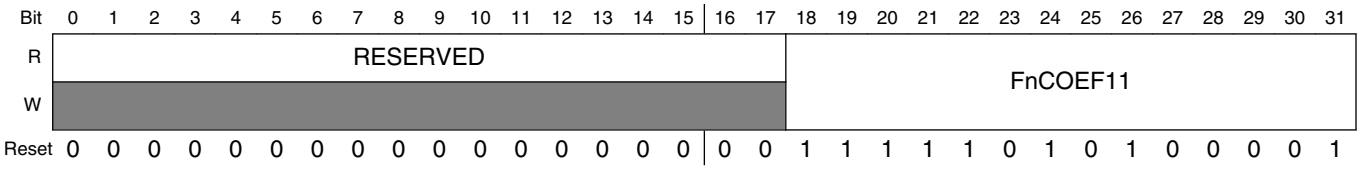
AFE_FLCOEF10*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF10	This field holds the value for coefficient 10 of decimation filter n (where n = 0 to 3).

48.3.40 Decimation Filter Coefficient Register 11 (AFE_FLCOEF11)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + ECh offset + (64d × i), where i=0d to 3d



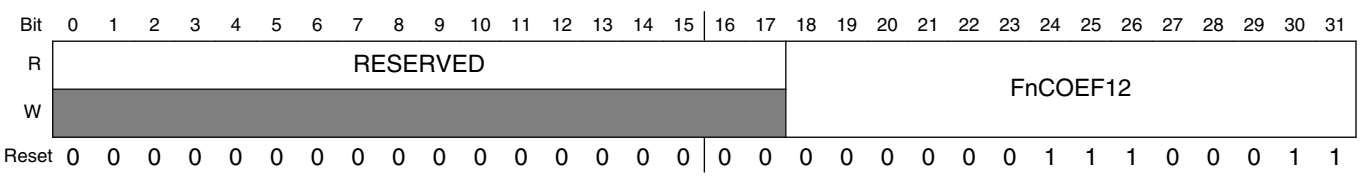
AFE_FLCOEF11*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF11	Decimation Filter n Coefficient 11. This field holds the value for coefficient 11 of decimation filter n.

48.3.41 Decimation Filter Coefficient Register 12 (AFE_FLCOEF12)

This holds one of the coefficients for decimation filter n. This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + F0h offset + (64d × i), where i=0d to 3d



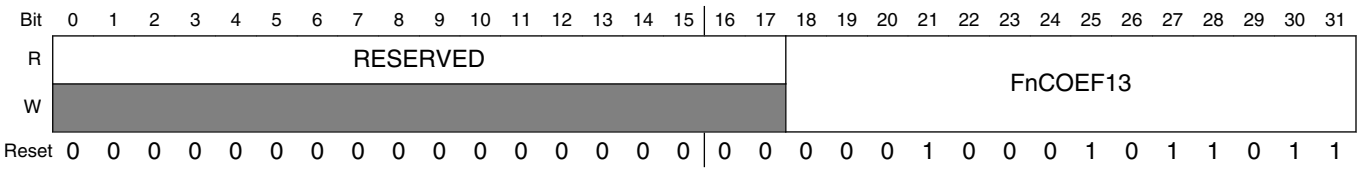
AFE_FLCOEF12*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF12	This field holds the value for coefficient 12 of decimation filter n (where n = 0 to 3).

48.3.42 Decimation Filter Coefficient Register 13 (AFE_FLCOEF13)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + F4h offset + (64d × i), where i=0d to 3d



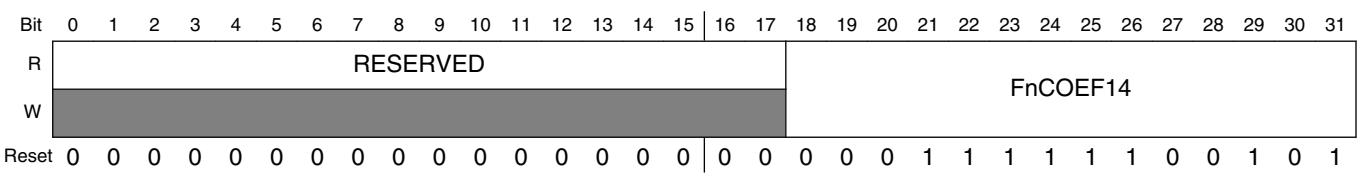
AFE_FLCOEF13*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF13	This field holds the value for coefficient 13 of decimation filter n (where n = 0 to 3).

48.3.43 Decimation Filter Coefficient Register 14 (AFE_FLCOEF14)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + F8h offset + (64d × i), where i=0d to 3d



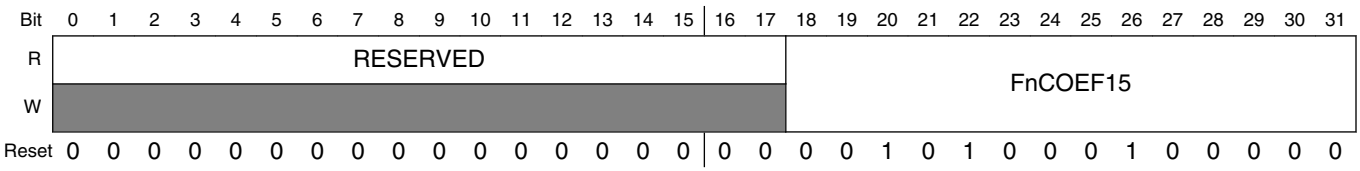
AFE_FLCOEF14*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF14	This field holds the value for coefficient 14 of decimation filter n (where n = 0 to 3).

48.3.44 Decimation Filter Coefficient Register 15 (AFE_FLCOEF15)

This holds one of the coefficients for decimation filter n (where n = 0 to 3). This register should not be changed while the filter enables from the CTE module are active. This register can only be written to using 16-bit or 32-bit accesses.

Address: 0h base + FCh offset + (64d × i), where i=0d to 3d



AFE_FLCOEF15*n* field descriptions

Field	Description
0–17 RESERVED	This field is reserved.
18–31 FnCOEF15	This field holds the value for coefficient 15 of decimation filter n (where n = 0 to 3).

48.4 Functional Description

48.4.1 Crystal Oscillator (XOSC)

The oscillator used on the AFE module is a low voltage, very low jitter 40MHz crystal oscillator. It features:

- Low RF emissions with peak to peak swing limited dynamically
- External crystal load capacitance is required
- Integrated internal bias resistor
- AC-coupled, non-hysteresis output buffer to provide very high immunity to jitter caused by power supply noise
- Clock monitor
- RMS jitter <1pS (100 Hz–5 MHz)
- External bypass mode
- Input 40 MHz only

This circuit is a loop amplitude controlled Pierce oscillator with programmable current settings. The optimum settings are determined in characterization, and have default settings that do not require adjustment. It is self-starting upon power-up. It is designed to work specifically with a 40MHz crystal with a Q>25,000. The oscillator's termination resistors are enabled by default.

48.4.2 Voltage Regulators (VREG)

Each of the voltage regulators in the AFE present low voltage detect flags to the system. In general the low voltage conditions do not immediately affect operation, but instead indicate an out of specification condition. VREG2 and VREG3, however, control the powerdown of the SDADC common mode circuit. If either of these two low voltage conditions is engaged, then the common mode circuitry will powerdown and the common mode voltage will decay to 0V and the SDADC's will be non-functional.

48.4.3 Sigma Delta Phase Locked Loop (SDPLL)

The SDPLL module is a calibrated, LC oscillator-based PLL generating only a 320 MHz clock. It features:

- Dual-path architecture
- LC oscillator with built-in inductor
- Self-calibration
- Non-programmable divider
- Phase Detector
- Lock detect
- Loss of reference clock detect
- It requires a 40 MHz reference clock

This circuit is an LC-based, calibrated PLL. It requires a 40 MHz reference, oscillates at 1.28 GHz, which is divided to the 320 MHz reference for the AFE. The oscillator is calibrated to be near the desired frequency by incrementally adding fixed capacitance to the oscillator and measuring the frequency. This allows a much lower oscillator frequency gain (Kv) than one which is not calibrated. The oscillator is thus much less sensitive to power supply or other noise. The calibration sequence is internally implemented, and occurs automatically when following the steps described in the Initialization section. The exact frequency is obtained by controlling the bias on voltage variable capacitors (VVC).

Dual path charge pumps are employed: a conventional (integral) charge pump to generate a frequency control voltage for the oscillator, and a feed forward (proportional) charge pump to reduce ripple and reference spurs on the control voltage.

48.4.4 Analog to Digital Converters (ADC's)

The 4 AFE ADC's are designed using a continuous time sigma delta modulator topology. The topology utilizes the accuracy of time to improve the accuracy in voltage by oversampling the input signal and filtering the results. The continuous time topology is used for the inherent benefits of built-in anti-aliasing filtering, immunity to substrate noise, and larger SNR over traditional Nyquist conversion at a less costly area and power budget comparatively. The topology requires a low jitter (less than 10 ps) 320 MHz clock and can achieve targeted SNR of 70 dB over a 5 MHz bandwidth. In addition, the topology of the continuous time ADC provide inherent anti-aliasing filtering. Because of this built-in AAF of the architecture there is no need for additional expensive active filtering in front of the ADC if a discrete time Nyquist rate or oversampled version of the ADC was implemented.

The sigma delta modulator topology is a fully differential cascade of integrators feedback form (CIFB) that utilizes active RC integrators to reduce sensitivities to layout parasitics. In addition, the feedback DAC currents of the ADC utilize NRZ DAC pulses to reduce the jitter requirements of the clock source. The sigma delta modulator along with the digital decimation filtering compose a complete ADC. The ADC described provides for final sampling rates from 1.25 Ms/sec to 10 Ms/sec with an effective accurate 12-bit resolution output.

48.4.5 Digital to Analog Converter (DAC)

The 12-bit DAC is designed for generating a ramp that is used to drive the VCO which is used to generate the frequency chirp for the RADAR system. The DAC uses current steering (steers current from negative side to positive side and vice versa) architecture to generate the analog step from digital input. The DAC's sample rate is 10MHz. The DAC is divided into two sub-DAC’s, one for the MSB’s which has 5-bits of input and the one for the LSB’s which has 7-bits of input. There is a binary to thermometer decoder for each section. The LSB section has 127 equally weighted current sources (ideal weight is 1uA for each current source). The MSB section has 31 equally weighted current sources (ideal weight should be 128uA for each current source). During the transition of all LSB current sources to one of the MSB sources (for example if the DAC digital input switches from 00000 1111111 to 00001 0000000) , the most significant current source

needs to be matched to the sum of all LSB current sources so that the DAC output increases only by 1uA. This is difficult to achieve with respect to power and area constraint.

The system can handle higher negative DNL(differential nonlinearity = (actual step-ideal step)/ideal step) i.e. the difference in output current can go negative during the LSB to MSB transition. So the DAC has been designed with an 12 LSB overlap between the LSB to MSB transitions throughout the DAC output range. The overlap is realized by setting the nominal value of the each MSB current sources to 116uA instead of the ideal 128uA.

Now due to the overlap, total loss of the current will be 12uA*31 = 372uA. In order to compensate for the loss two additional MSB current sources are added (2*116uA = 232uA, still DAC will have 140uA loss which will be compensated for by factory trim settings applied at device reset . This extra two MSB sources are controlled by DACCTRL[FS_ADJ].

Every current source inside the DAC is connected to a dedicated pair of switches S1 and notS1, all of the S1 switches connect the current sources to DAC_OUTP and are controlled by DATA (DAC digital inputs) and all of the notS1 switches connect the current sources to DAC_OUTN and are controlled by notDATA . Now, if DACCTRL[FS_ADJ] bits are both set to logic “low” then the 232uA will flow to the DAC_OUTN. If one of the FS_ADJ bits is set to logic “low” and the other one logic “high” then the one set to logic “high” will provide 116uA to the DAC_OUTP and the other one (set to logic “low”) will provide 120uA to the DAC_OUTN. If both FS_ADJ bits are set to logic “high” then the 232uA will flow to the DAC_OUTP.

48.4.6 Decimation Filters

48.4.6.1 Introduction

The AFE radar sigma-delta decimation filter is a digital block that low-pass filters and decimates the 320MHz 5-bit sigma-delta modulator output to generate a lower rate 14-bit signed two’s complement output. The decimation filter supports oversampling ratios (OSR) from 32 to 256 to produce a 10 to 1.25 MHz output sample rate. It includes two stages of filtering: a fourth order sinc filter followed by a programmable 32-tap FIR filter as shown in Figure 48-5. The sinc filter down samples the input by a factor of OSR/4 with the final decimate by 4 done by the FIR filter. The sinc filter output is scaled to a 14-bit signed value for filtering by the FIR filter. The FIR filter has programmable coefficients which are set to yield the desired overall frequency response and digital gain.

After FIR filtering, digital limiting is performed to clip signals with amplitudes greater than the full scale digital output. The output (filt_out[13:0]) is a signed 14-bit, two's complement value that can be optionally rounded to 12-bits.

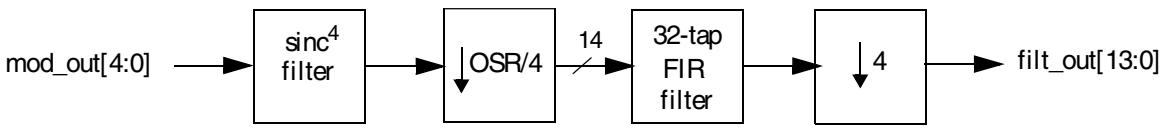


Figure 48-5. Decimation Filter Block Diagram

Table 48-1. OSR Selection

FILTnCTRL[OSR]	OSR	Sample Rate (320 MHz)
00	32	10
01	64	5
10	128	2.5
11	256	1.25

48.4.6.2 Filter Parameters

The combined effects of the sinc filter and the default FIR filter coefficient values are shown in Table 48-2

Table 48-2. Filter Parameters

filt_osr[1:0]	OSR
Modulator Frequency	320 MHz
Output Sample Rate (Fs)	10 - 1.25 MHz
Output Resolution	12 / 14 bits
FIR Filter Length	32
FIR Coefficient Size	14 bits
OSR	32 - 256
Pass-band	0 - 0.4 Fs
Pass-band Ripple	< +/- 0.3 dB[1]
Stop-band Attenuation > 0.7 Fs	> 60 dB
Phase Response	Linear
Group Delay	4.3 / Fs

1. When using 320 MHz sampling mode.

48.4.6.3 Control and Operation

The filter state is reset and filtering operations are stopped when reset is asserted. Out of reset, the filter is disabled until the filt_en signals from the CTE are asserted. When the filt_en signals are negated, the state of the filter is reset and the filter effectively enters a low power mode by negating the enables to most registers allowing for internal clock gating of the registers. The FIR filter coefficient values are not reset and can be read/ written when the filt_en signals are negated.

The FIR filter coefficients have default values which give the filter the characteristics shown in Table 48-2 and described in Frequency Response . The coefficients are programmable and can be modified. The FIR filter is symmetrical with FLnCOEF0 corresponding to FIR coefficients h[0] and h[31], FLnCOEF1 corresponding to FIR coefficients h[1] and h[30], up to FLnCOEFF corresponding to coefficients h[15] and h[16]. Transfers take one clock cycle and may be performed on consecutive clock cycles.

After the coefficients are programmed and the OSR is selected, operation of the filter is started by the AFE_FILT_PHASE block of the MISC module asserting the filt_en signals. This assertion establishes the sampling phase for the channels. The timing of the filt_en signals for each channel can be used to synchronize sampling of multiple channels or introduce a phase delay between channels. The coefficients, OSR, and rounding mode should not be modified while the filter is operating.

The mod_out data is sampled on the rising edge of mod_clk. This 5-bit signed, two's complement input signal is filtered by the sinc filter which has a 29-bit internal word width. The sinc filter output it scaled based on the OSR and rounded to produce a 14-bit signed output which further processed by the FIR filter.

The FIR filter is a symmetrical with 16 programmable coefficients that are 14-bit signed, Two's complement, fractional values which can range in value from -1.0 (0x2000) to 0.9999 (0x1FFF). The FIR filter is implemented with a full 14-bit multiplier and 29-bit accumulator. The FIR filter accumulator has two guard bits which provide headroom for large amplitude signals. The output of the FIR filter is limited and rounded to produce a 12-bit two's complement output with the 2 least significant bits always zero. The pass-band gain of the filter establishes the digital gain which sets the full scale analog input level. A gain of 1.22 (1.7dB) is required to produce the specified full scale analog input range. When the FIR filter processing is finished, the conversion complete (filt_cc) output is asserted for one clock cycle to indicate the filt_out[13:0] value is valid. The first 8 samples generated after the filter is enabled are not valid due to the memory of the filter. The FIR filter processing adds 55 mod_clk (320MHz) clock cycles of delay to the filter, in addition to the filter group delay (4.3 sample times for the default coefficients).

48.4.6.4 Frequency Response

The filter frequency response scales with the output sample rate which is a function of the modulator clock (mod_clk) frequency and the OSR. The overall frequency response is the combined response of the sinc4 filter and the 32 tap FIR filter. The sinc4 filter frequency response which is given in equation 1-1 has significant pass-band droop and a wide transition band.

$H(z)=((1-z^{-OSR}) /(1-z^{-1}))^4$

An example of the $sinc^4$ filter frequency response with OSR=32 is shown in the figure below. In this, case the sample frequency is 10 MHz, so frequency components above 5 MHz are aliased back into the band from 0 to 5 MHz.

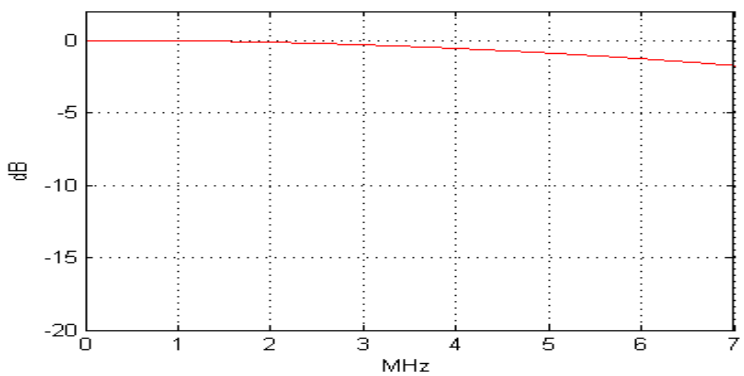


Figure 48-6. Sinc Filter Frequency Response OSR=32

The FIR filter is used to compensate for the pass-band droop of the sinc filter and narrow the transition band. The specifications for the combined response of the two filters is given in Filter parameters table and shown in the following figure for the default FIR filter coefficients.

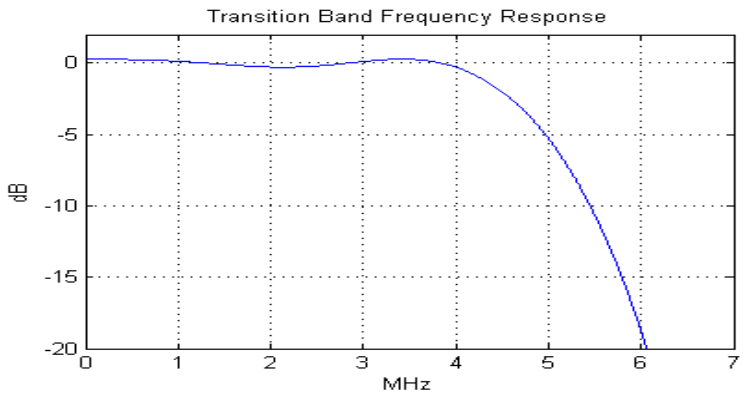


Figure 48-7. Overall Filter Frequency Response OSR=32

48.5 Resets

All wrapper registers are forced to their reset state upon the assertion of the reset input to the AFE wrapper.

48.6 Clocks

The SDPLL in the AFE analog block generates 320 MHz, 160 MHz, and 80 MHz outputs via non-configurable dividers. These clocks are used in the wrapper by the decimation filters. The analog block also produces a low jitter 40 MHz crystal reference clock.

48.7 Interrupts

The AFE wrapper has several interrupts. All flags are sticky and will remain set. Flags are cleared by writing a 1 to that bit position except for SDPLL lock flag and loss of reference flag.

Table 48-3. AFE Interrupt Sources

Interrupt Source/Flag	Local Enable	Description
ADCOVLD[OVERVOLT]	ADCCTRL7[OVLIE]	ADC input overvoltage detected. One combined interrupt for all ADC's.

Table continues on the next page...

Table 48-3. AFE Interrupt Sources (continued)

Interrupt Source/Flag	Local Enable	Description
OSCSTS[STS]	OSCCTRL[IE]	Oscillator status. Only activated by rising edge of STS bit.
PLLSTS[LOR]	PLLCTRL3[LORIE]	SDPLL loss of reference clock.
PLLSTS[LOCK]	PLLCTRL3[LCKHIIE]	SDPLL lock achieved.
PLLSTS[LCKLOSS]	PLLCTRL3[LCKLOIE]	SDPLL lock lost.

Chapter 49
MIPICSI2

49.1 Chip specific MIPICSI2 information

49.1.1 CSI signals pin assignment

The following table provides the device IO pin mapping.

Table 49-1. CSI signals pin assignment

Signal description	Pin name
CKP (clock lane differential positive signal)	CSI_CLKP
CKN (clock lane differential negative signal)	CSI_CLKN
DP0 (data lane 0 differential positive signal)	CSI_LANE0P
DN0 (data lane 0 differential negative signal)	CSI_LANE0N
DP1 (data lane 1 differential positive signal)	CSI_LANE1P
DN1 (data lane 1 differential negative signal)	CSI_LANE1N
DP2 (data lane 2 differential positive signal)	CSI_LANE2P
DN2 (data lane 2 differential negative signal)	CSI_LANE2N
DP3 (data lane 3 differential positive signal)	CSI_LANE3P
DN3 (data lane 3 differential negative signal)	CSI_LANE3N
REXT (to connect to external reference resistor for auto-calibration)	REXT

49.2 About this module

49.2.1 Definition

MIPICSI2 is a hardware module that:

- Provides a low-cost, high speed, serial receive interface between a MIPICSI2 serial input data and the on chip Video Interface Unit (VIU) high speed radar data and signal processing tool box(SPT).
- Incorporates a four lane physical layer and one clock lane, compliant with the MIPI Alliance Standard for D-PHY. DPHY RX is high-frequency, low-power, low-cost, and source-synchronous.
- Incorporates a receiver controller core that is flexible, high-performance, easy-to-use and MIPICSI2 compliant. It interfaces with DPHY RX on one side and the VIU Signal Processing Toolbox (SPT) module at the user interface side.
- Incorporates a VIU gasket.

49.2.2 MIPICSI2 Copyright

49.2.3 Features

MIPICSI2 subsystem features include:

- Support for up to four lanes on the DPHY RX side for data transfer.
- Data rate up to 1 Gbit/s1 Gbit/s . supported on each individual DPHY RX lane. It is expected that the throughput when all lanes are being used will not exceed 1Gb/s.
- Short and long packet decoding.
- Support for generic short packet data types for passing any timing related information from the transmitter side to the software.
- Frame and line synchronization packet signaling.
- Input side data types: RGB888, RGB666, RGB565, YUV422 8- and 10-bit, RAW8, RAW10, RAW12, RAW14, user defined, embedded
- Output side data types: RGB888, RGB666, RGB565, YUV444 8-bit
- 1 pixel output per user interface clock.
- Error management and handling:
  - DPHY level errors
  - Packet level errors
  - Protocol decoding level errors

all of which are MIPICSI2 specification compliant with *MIPI Alliance D-PHY Specification* and *MIPI Alliance CSI-2 Specification*, v01-01-00 by MIPI® Alliance.

49.2.4 MIPICSI2 compliance

MIPICSI2 module complies with the MIPI® Alliance Specification for MIPICSI2 (Version 1.01.00, 9-Nov-2010), available from the MIPI Alliance.

49.2.5 Modes of operation

MIPICSI2 supports the following mode of operation:

Mode	Description
Normal	MIPICSI2 subsystem operates as an interface between the external video dataradar data and the Video Interface Unit (VIU)Signal Processing Toolbox (SPT) module.

49.2.6 Clocking

The table below lists the frequency requirements on the clocks for this module.

Table 49-2. Clock Description

Clock Name	Frequency Requirement(MHz)
controller_li_clk	Should be greater than equal to lane frequency /8
controller_ui_clk	Should be greater than equal to 2/3 x controller_li_clk
DPHY_ESCAPE_CLK	60(Min) 80(Max)

49.3 MIPICSI2

49.3.1 MIPICSI2 block diagram

This diagram below shows the components of MIPICSI2:

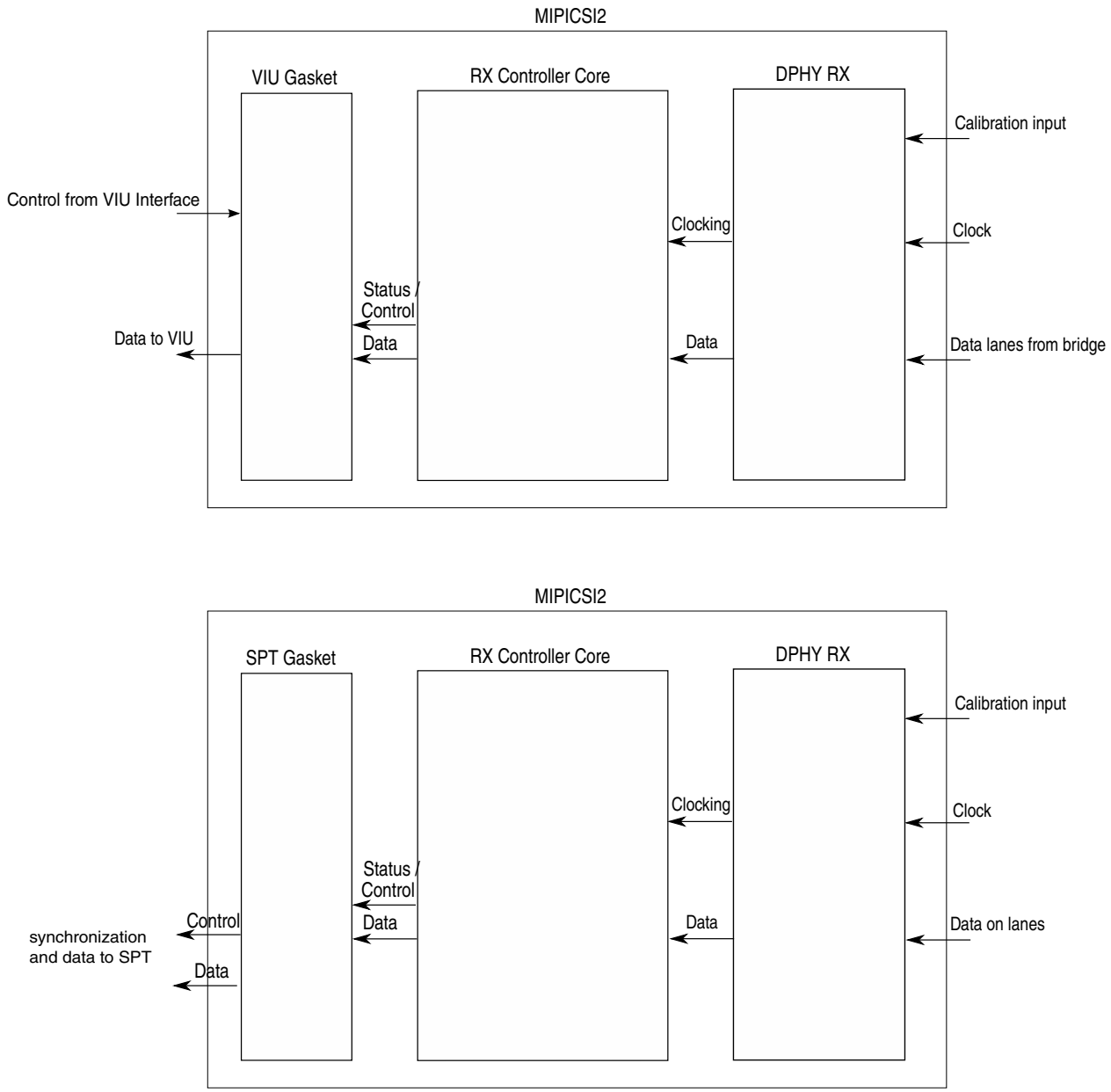


Figure 49-1. MIPICSI2 block diagram

49.3.2 MIPICSI2 components

This table below describes the functions of the main components of the MIPICSI2 subsystem:

Component	Function
DPHY RX	Receives off-chip data over the MIPICSI2 compliant physical layer interface, de-serializes the radar data in to bytes, and sends the bytes to the RX controller core.
RX controller core	Receives MIPICSI2 packetized data as bytes from DPHY RX, provides lane management and low level protocol functionality, and then sends the de-packetized pixels to the VIU gasket.
VIUSPT gasket	Receives the de-packetized data from the RX controller core and sends it out with associated synchronization and blanking information to the Video Interface Unit (VIU)SPT module.

49.3.3 MIPICSI2 signals

This table below describes the signals on the boundary of the MIPICSI2 subsystem:

Signal	Function	Direction	Level of signal after reset	Clocking
PHY interface				
Calibration input	Pin to connect an external reference resistor for auto-calibration. 15K ohm precise resistor with 1% variation or lower is required.	Input/Output	—	—
Clock	Receives the differential clock.	Input	—	—
Data lanes	Receives the differential data.	Input	—	—
Pixel data	Provides the formatted pixel data through a 24-bit parallel interface to the VIU module.	Output	000000h	controller_ui_clk
Sync and enable	Provides a horizontal and vertical synchronization pulses and a data enable signal to the VIU module.	Output	1	controller_ui_clk

49.4 DPHY RX

49.4.1 DPHY RX block diagram

This diagram below shows the components of DPHY RX:

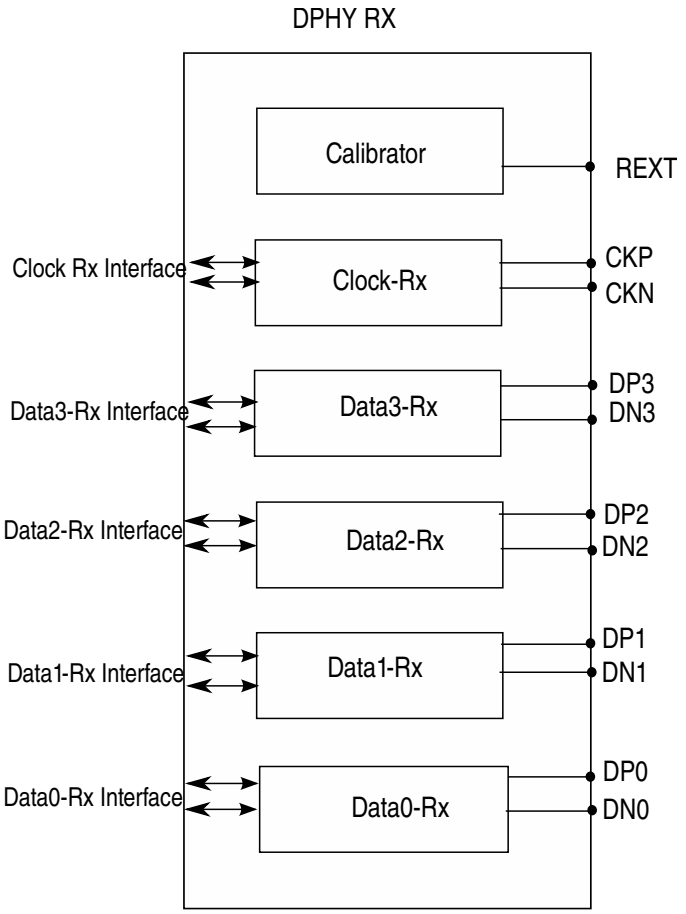


Figure 49-2. DPHY RX block diagram

NOTE

The DPHY RX used on this chip consists of four data lanes.

49.4.2 DPHY RX components

This table below describes the functions of the main components of DPHY RX:

Component	Function
Data0-Rx Data1-Rx Data2-Rx Data3-Rx	Receives off chip data through unidirectional receiver lane. Consists of a high-speed receiver (HS-RX), a low-power receiver (LP-RX), a low-power contention detector (LP-CD), and a deserializer.
Clock_Rx	Receives off chip clock through unidirectional receiver lane and consist of a high-speed receiver (HS-RX), a low-power receiver (LP-RX), and a low-power contention detector (LP-CD).
Calibrator	Calibrates the receiver termination after the block is powered-up to ensure that the transmitter and receiver meet the required specifications. Runs in auto-calibration mode or user programmable mode.

49.4.3 DPHY RX signals

The table below described the DPHY RX signals:

Table 49-3. DPHY RX signals

Signal	Function	Direction
CKP	Receives the differential positive clock lane.	Input
CKN	Receives the differential negative clock lane.	Input
DP0	Receives the differential positive data lane 0.	Input
DN0	Receives the differential negative data lane 0.	Input
DP1	Receives the differential positive data lane 1.	Input
DN1	Receives the differential negative data lane 1.	Input
DP2	Receives the differential positive data lane 2.	Input
DN2	Receives the differential negative data lane 2.	Input
DP3	Receives the differential positive data lane 3.	Input
DN3	Receives the differential negative data lane 3.	Input
REXT	Pin to connect an external reference resistor to for auto-calibration. An accurate 15K ohm resistor with 1% variation or lower is required.	Input/Output

NOTE

Please refer to IO Signal Description Input Multiplexing sheet (Excel file) attached to this document for external signals multiplexing.

49.4.4 Calibrator

Calibrator is used to calibrate the RX termination.

DPHY RX supports one of the following receiver termination modes:

Table 49-4. DPHY RX supports description

Mode	Description
Auto calibration	Termination resistors are automatically configured by calibration circuit with reference resistor connected between REXT and VSSA. NOCAL field is set to zero for such purposes.
User programmable	NOCAL is set to one for manual calibration. RCALI is programmed by the user through registers.

When calibration completes, CALCOM is set indicating that the calibration is complete.

Table 49-5. Calibrator specifications

Symbol	Parameter	Min	Typ	Max	Unit
R_{EXT}	External reference resistor, 1% accuracy, for auto-calibration.		15		KΩ
T_{cal}	Time from when PD_RX signal goes low to when CALCOM goes high		2		µs

49.4.5 Receiver

The receiver blocks within the DPHY RX module consists of unidirectional receive (RX) data and clock lanes.

The RX data lanes consist of:

- a High Speed Receiver
- a Low-Power Receiver
- a Low-Power Contention Detector
- a Deserializer

The clock lane consists of:

- a High Speed Receiver
- a Low-Power Receiver
- a Low-Power Contention Detector

The high-frequency signals have a low voltage swing, whereas low-power signals have a large voltage swing. The high speed function is used for high-frequency data traffic, and the low-power functions (10 Mbit/s data rate) are typically used for control commands.

After the MIPICSI2 subsystem has been initialized, as described in Initializing the MIPICSI2 subsystem, the data lanes are ready to receive low-power data. The transmitter side needs to drive stop state on the lanes for a predetermined time, indicated by the clock and data lanes having stop state asserted which completes the initialization of the link. The steps required to place the receiver in to high-speed and ultra low-power modes are described in Place receiver in high-speed mode and Place receiver in ultra low-power mode.

49.5 RX controller core

49.5.1 RX controller core block diagram

This diagram below shows the components of RX core controller:

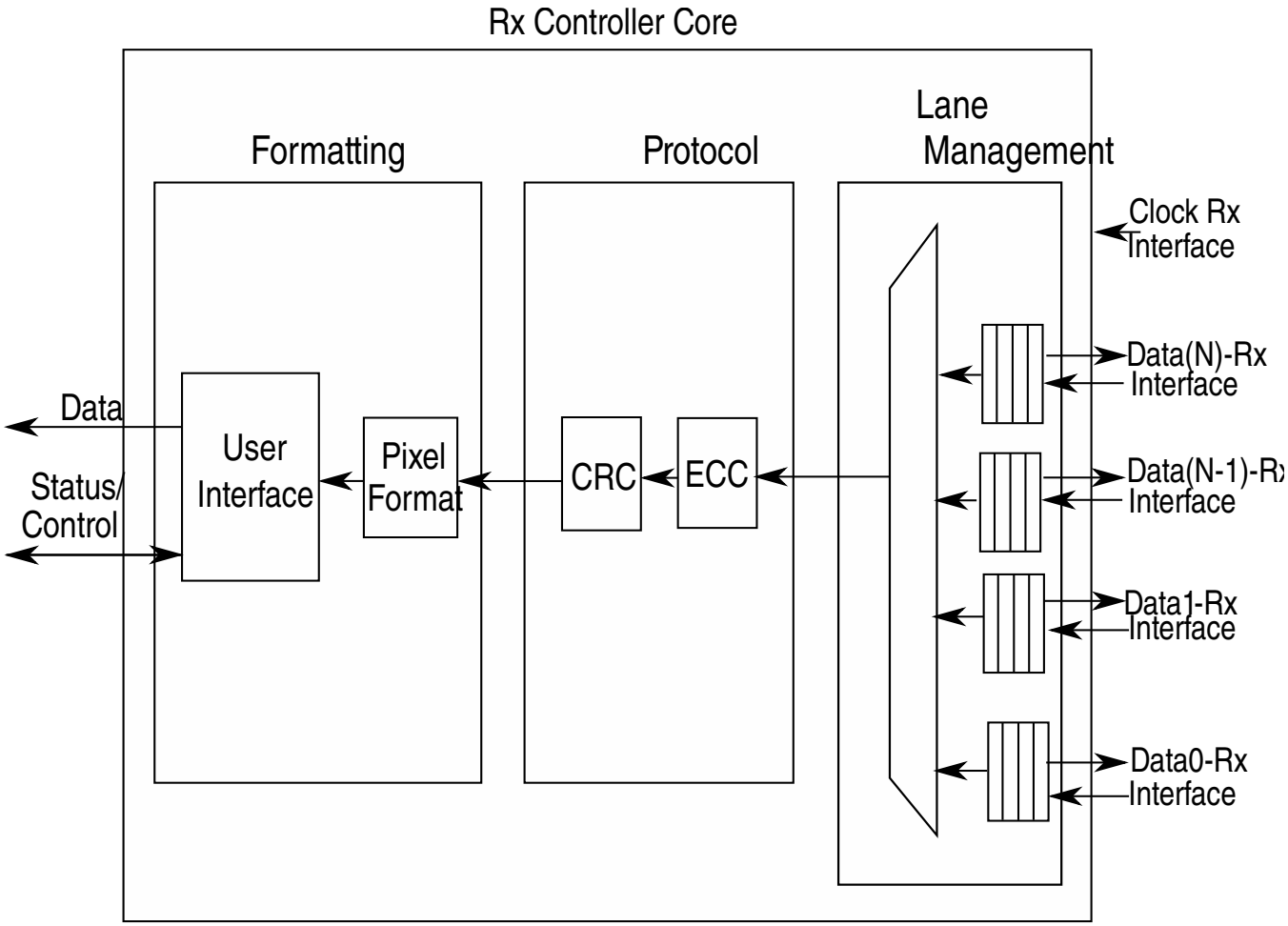


Figure 49-3. RX controller core block diagram

NOTE

The RX controller core used on this chip consists of four data lanes.

49.5.2 RX controller core components

This table below describes the functions of the main components of RX core controller:

Component	Function
Lane management	Controls lane management as defined by the MIPICSI2 specification.
Protocol	Controls low level protocol, including error handling, as defined by the MIPICSI2 specification.
Formatting	Controls byte to pixel packing as defined by the MIPICSI2 specification.

49.5.3 RX controller core signals

This table below describes the RX controller core signals:

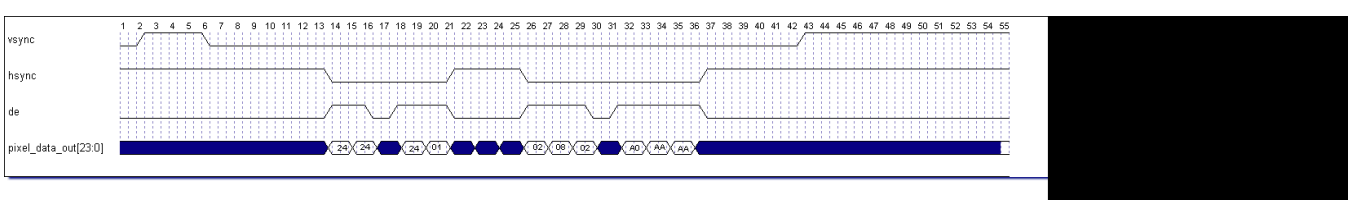
Table 49-6. RX controller core signals

Signal	Function	Direction
Data0-Rx Interface	Receives the output from Data0-Rx of DPHY RX.	Input
Data1-Rx Interface	Receives the output from Data1-Rx of DPHY RX.	Input
Data2-Rx Interface	Receives the output from Data2-Rx of DPHY RX.	Input
Data3-Rx Interface	Receives the output from Data3-Rx of DPHY RX.	Input
Clock Rx Interface	Detects any lane transitions in low power mode.	Input
Status/control	Provides status and control information to the VIU gasket.	Input/Output
Data	Provides the formatted pixel data to the VIU gasket.	Output

49.6 VIUSPT gasket

49.6.1 MIPICSI2 subsystem output data format

This timing diagram shows example signals from the output of the MIPICSI2 subsystem from the Video Interface Unit (VIU) gasket state machine:



The below table shows the organization of image data at the output of the MIPICSI2 subsystem:

Table 49-7. MIPICSI2 subsystem output data

Data formats	Bit Wise Alignment																							
—	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW 8-bit	0	0	0	0	0	0	0	0	0	0	sample[7:0]								0	0	0	0	0	0
RAW 10-bit	0	0	0	0	0	0	0	0	0	0	sample[9:0]										0	0	0	0
RAW 12-bit	0	0	0	0	0	0	0	0	0	0	sample[11:0]												0	0
RAW 14-bit	0	0	0	0	0	0	0	0	0	0	sample[13:0]													
Embedded	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	sample[7:0]								
User Defined Data	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	sample[7:0]								
YUV 422 8-bit	0	0	0	0	0	0	0	0	Y1[7:0]							U1[7:0]								
-	0	0	0	0	0	0	0	0	Y2[7:0]							V1[7:0]								
YUV 422 10-bit	0	0	0	0	Y1[9:0]										U1[9:0]									
-	0	0	0	0	Y2[9:0]										V1[9:0]									
RGB565	R[4:0]					0	0	0	0G[5:0]						0	0	B[4:0]				0	0	0	
RGB888	R[7:0]								G[7:0]								B[7:0]							

This table shows the output data format for RGB888 type data:

Pixel out at the controller interface pixel_data_out [23:0]			Pixel out at the controller interface pixel_data_out [23:0]		
[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]
{red[7:0]}	{green[7:0]}	{blue[7:0]}	{red[7:0]}	{green[7:0]}	{blue[7:0]}

This table shows the output data format for RGB888 type data:

Pixel out at the controller interface pixel_data_out [23:0]			Pixel out at the controller interface pixel_data_out [23:0]		
[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]
{red[7:0]}	{green[7:0]}	{blue[7:0]}	{red[7:0]}	{green[7:0]}	{blue[7:0]}

This table shows the output data format for RGB666 type data:

Pixel out at the controller interface pixel_data_out [23:0]			Pixel at the VIU interface pixel_pixel_data_out [23:0]		
[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]
{6'b0,red[5:4]}	{red[3:0],green[5:2]}	{green[1:0]] , blue[5:0]}	{red[5:0],2'b00}	{green[5:0],2'b00}	{blue[5:0],2'b00}

This table shows the output data format for RGB565 type data:

Pixel out at the controller interface pixel_data_out [23:0]			Pixel at the VIU interface pixel_pixel_data_out [23:0]		
[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]

Table continues on the next page...

Pixel out at the controller interface pixel_data_out [23:0]			Pixel at the VIU interface pixel_pixel_data_out [23:0]		
{8'b0}	{red[4:0],green[5:3] }	{green[2:0]] , blue[4:0]}	{red[4:0],3'h0}	{green[5:0],2'b00}	{blue[4:0],3'h0}

This table shows the output data format for YUV422 8-bit type data:

Pixel number	Pixel out at the controller interface data_out[23:0]			Pixel out at the controller interface data_out[23:0]		
	[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]
1	8'b0	Y1	U1	—	—	—
2	8'b0	Y2	V1	Y1	U1	V1

This table shows the output data format for YUV422 10-bit type data:

Pixel number	Pixel out at the controller interface data_out[23:0]			Pixel out at the controller interface data_out[23:0]		
	[23:16]	[15:8]	[7:0]	[23:16]	[15:8]	[7:0]
1	{4'b00,Y1[9:6]}	{Y1[5:0],U1[9:8]}	U1[7:0]	—	—	—
2	{4'b00,Y2[9:6]}	{Y2[5:0],V1[9:8]}	V1[7:0]	Y1[9:2]	U1[9:2]	V1[9:2]

49.7 Using MIPICSI2

49.7.1 Initializing the MIPICSI2 subsystem

To enable the MIPICSI2 subsystem after the Power supply/connection to the MIPI Receiver has been disconnected or chip has been reset:

1. Configure the number of data lanes (NULANE).
2. Bring the receiver module in DPHY RX out of reset (PDRX).
3. Enable the clock and data lanes of the DPHY (RXEN).
4. Wait until auto calibration is completed (CALCOM) and the DPHY RX is ready to monitor the low-power data from the transmitter, usually 2 us after DPHY RX exits reset.
5. Write the required settle time (HSSETL) and the DPHY lane high speed receive settle time (D0HSET). The settle time should be calculated as described in Calculate the required settle time for DPHY RX.

49.7.2 Calculate the required settle time for DPHY RX

The time in which the high-speed receiver will ignore any high-speed transitions on the data lane is dependent on DPHY_ESCAPE_CLK frequency and the data rate.

1. Calculate the required settle time using the formula **Settle time = (HSSETL + 1) × (Tperiod of DPHY_ESCAPE_CLK)**, where DPHY_ESCAPE_CLK is 80 MHz (12.5ns).

The settle time value varies in the range 85+6×UI (minimum) to 145 +10×UI (maximum) as per the formula shown in the table below.

For example, keeping data rate as 1 Gb/s , the unit interval (UI) is 1ns.

Taking a mid value of settle time (120 ns) the value of HSSETL must be programmed as:

Table 49-8. Settle time for DPHY RX

	Minimum	Typical	Maximum
T_{HS_SETL}	85ns + 6×UI	120ns	145ns +10×UI
HSSETL	(91/15.04) -1 = 5	(120/15.04) -1 = 6.97 ~ 7	(155/15.04)-1 = 9.3058 ~9
HSSETL	(91/12.5) -1 = 6	(120/12.5) -1 = 8.6 ~ 9	(155/12.5)-1 = 11

49.7.3 Override auto calibration values

Termination resistors are automatically configured by the calibration circuit with a reference resistor connected between REXT and VSSA. The auto calibration procedure is performed every time DPHY RX is powered-up.

To override the auto calibration values and enter another value in user programmable mode:

1. Enter user programmable mode (NOCAL).
2. Override the calibration value (RCALI).
3. Wait until calibration has completed (CALCOM).

49.7.4 Place receiver in high-speed mode

After DPHY RX has been initialized for receiving data, the data lanes are ready to receive low-power data.

The transmitter side needs to drive the stop state on the lanes for a predetermined time. High-speed receive is enabled by the transmitter driving the high-speed command on the respective lane, as per MIPI D-PHY standard *LP11-LP01-LP00*.

The following steps are performed to receive high-speed data:

1. Once in the bridge state (LP00), the receiver side will wait for the programmed settle time, HSSETL before it starts to monitor the data lane.
2. RXACTH indicates that high-speed transmission is active on the data lane.
3. At the start of a high speed transmission, the receiver looks for the synchronization pattern sent by the transmitter.
4. When DPHY RX has received one byte of deserialized data, it sends it to RX controller core and updates RXVALH indicating that the current data is valid.

Once a complete packet of high-speed data has been received the lanes return to stop state.

49.7.5 Place receiver in ultra low-power mode

Ultra low-power (ULP) mode is entered through a sequence of low-power commands from the transmitter on the lane.

DPHY RX reacts to the command sequence in the following way:

- DPHY RX indicates that ULP mode is active on the lane once the command sequence is complete (ULPSC).
- DPHY RX indicates that ULP mode is exited when CULPMA is cleared.

Once ULP mode has been exited the lanes return to stop state.

49.7.6 Understanding MIPICSI2 compliant error levels

Due to the layered structure of the MIPICSI2 subsystem, errors can occur at different levels. This table described the levels.

Table 49-9. MIPICSI2 compliant error levels

Error Level	[1]Description
DPHY	Any DPHY related transmission error. Unrelated to the transmission's content. The recommended behavior for handling this error level covers only those errors generated by the data lane since an implementation can assume that the clock lane is running reliably as provided by the expected bit-error-rate (BER) of the link.
Packet	Data integrity of the received packet header and payload data. The behavior for this error level covers only errors recognized by decoding the packet header's ECC byte and computing the CRC of the data payload.

Table continues on the next page...

Table 49-9. MIPICSI2 compliant error levels (continued)

Error Level	[1]Description
Protocol Decoding	Errors present in the decoded packet header or errors resulting from an incomplete sequence of events. The recommended behavior for this error level covers errors caused by decoding the packet header information and detecting a sequence that is not allowed by the MIPICSI2 protocol or a sequence of detected errors by the previous layers.

1. Copyright © 2005-2010 MIPI Alliance, Inc. All rights reserved.

49.7.7 Decipher error sources and interrupt signals

The MIPICSI2 subsystem provides an error based interrupt signaltwo MIPI protocol based errors.[1]. This interrupt can have a variety of sources.

Table 49-10. Decipher error sources and interrupt signals

Source of error		Details	Error Level	Associated ERRPPST field	Associated ERRPHYS field[1]
Start-of-Transmission (SoT)		A recoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted, but in such a way that proper synchronization can still be achieved. This is considered to be a “soft error” in the leader sequence and confidence in the payload data is reduced.	DPHY	—	
SoT synchronization		An unrecoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted in a way that proper synchronization cannot be expected.	DPHY		or
Control		This error is asserted when the DPHY has detected an illegal control sequence, and remains high until the next change in line state. For example, if a turn-around request or escape mode request is immediately followed by a stop state instead of the required bridge state.	DPHY	—	or
Double bit-error		Asserted when an ECC syndrome was computed and two bit-errors are detected in the received header.	Packet		—
Single bit-error		Asserted when an ECC syndrome was computed and a single bit-error in the header was detected and corrected. Error Position (MIPICSI2_ERRPOS) register contains the location where the single bit error was detected, but is only valid if is 1.	Packet		—

Table continues on the next page...

1. Copyright © 2005-2010 MIPI Alliance, Inc. All rights reserved.

Table 49-10. Decipher error sources and interrupt signals (continued)

Source of error		Details	Error Level	Associated ERRPPST field	Associated ERRPHYS field[1]
Packet payload error		Asserted when the computed CRC code of the message payload is different than the received CRC code.	Packet		—
Frame sync[2]		Asserted when Frame End (FE) is not paired with Frame Starts (FS).Several error cases can be identified for this type of error, such as, when a FS is followed by a second FS, the frame corresponding to the first FS is considered an error,or an FE is followed by a second FE, or when a packet level double bit-error was signaled from the protocol layer the whole transmission until the first DPHY stop state should be ignored since it contains no information that can be safely decoded and cannot be qualified with a data valid signal.	Protocol Decoding		—
Unrecognized ID		Unimplemented or unrecognized ID in the header. Invalid ID Report Register (MIPICSI2_INVID) reports invalid ID was found and what the invalid ID was.	Protocol Decoding		—
Data payload received between FS and FE contains errors		This signal is asserted on any packet payload error and de-asserted on the first FE.	Protocol Decoding		—

1. One field per DPHY RX data lane
2. SoT synchronization also generates this error.

At reset all sources are masked in Interrupt Enable (MIPICSI2_INTREN). Reading the error registers after the interrupt signal is raised will provide the source.

49.7.8 Interrupts

The MIPICSI2 subsystem provides interrupt output that can be error based or status based. A total of one three interrupts can be triggered by the MIPICSI2 subsystem.

1.General MIPICSI2 protocol based interrupt- One interrupt for the protocol based errors and status

- Error based interrupt - This interrupt can have a variety of sources as described in the table below :

Table 49-11. General MIPICSI2 protocol Error based interrupt

Source of error		Details	Error Level	Associated ERRPPST field	Associated ERRPHYS field[1]
Start-of-Transmission (SoT)		A recoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted, but in such a way that proper synchronization can still be achieved. This is considered to be a “soft error” in the leader sequence and confidence in the payload data is reduced.	DPHY	—	
SoT synchronization		An unrecoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted in a way that proper synchronization cannot be expected.	DPHY		or
Control		This error is asserted when the DPHY has detected an illegal control sequence, and remains high until the next change in line state. For example, if a turn-around request or escape mode request is immediately followed by a stop state instead of the required bridge state.	DPHY	—	or
Double bit-error		Asserted when an ECC syndrome was computed and two bit-errors are detected in the received header.	Packet		—

Table continues on the next page...

Table 49-11. General MIPICSI2 protocol Error based interrupt (continued)

Source of error		Details	Error Level	Associated ERRPPST field	Associated ERRPHYS field[1]
Single bit-error		Asserted when an ECC syndrome was computed and a single bit-error in the header was detected and corrected. Error Position (MIPICSI2_ERRPOS) register contains the location where the single bit error was detected, but is only valid if is 1.	Packet		—
Packet payload error		Asserted when the computed CRC code of the message payload is different than the received CRC code.	Packet		—
Frame sync[2]		Asserted when Frame End (FE) is not paired with Frame Starts (FS) on a given virtual channel (VC).Several error cases on the same VC can be identified for this type of error, such as, when a FS is followed by a second FS on the same virtual channel the frame corresponding to the first FS is considered an error, or when a packet level double bit-error was signaled from the protocol layer the whole transmission until the first DPHY stop state should be ignored since it contains no information that can be safely decoded	Protocol Decoding		—

Table continues on the next page...

Table 49-11. General MIPICSI2 protocol Error based interrupt (continued)

Source of error		Details	Error Level	Associated ERRPPST field	Associated ERRPHYS field[1]
		and cannot be qualified with a data valid signal.			
Unrecognized ID		Unimplemented or unrecognized ID in the header. Invalid ID Report Register (MIPICSI2_INVID) reports the VC on which the invalid ID was found and what the invalid ID was.	Protocol Decoding		—
Data payload received between FS and FE contains errors		Asserted after FE when the data payload received between FS and FE contains errors. This signal is asserted on any packet payload error and de-asserted on the first FE.	Protocol Decoding		—

1. One field per DPHY RX data lane
2. SoT synchronization also generates this error.

- Status based interrupt - This interrupt is triggered on any protocol based status update. After reset all sources are masked. Reading the relevant status register will provide the source. The sources are listed below.
  - Generic short packet received

1.General MIPICSI2 PHY based interrupt- One interrupt for the PHY based errors

- Error based interrupt - This interrupt can have a variety of sources as described in the table below :

Table 49-12. General MIPICSI2 PHY Error based interrupt

Source of error		Details	Error Level	Associated ERRPHYS field[1]
Start-of-Transmission (SoT)		A recoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted, but in such a way that proper synchronization can still be achieved. This is	DPHY	

Table continues on the next page...

Table 49-12. General MIPICSI2 PHY Error based interrupt (continued)

Source of error		Details	Error Level	Associated ERRPHYS field[1]
		considered to be a “soft error” in the leader sequence and confidence in the payload data is reduced.		
SoT synchronization		An unrecoverable error that is asserted for one cycle if the high-speed SoT leader sequence is corrupted in a way that proper synchronization cannot be expected.	DPHY	or
Control		This error is asserted when the DPHY has detected an illegal control sequence, and remains high until the next change in line state. For example, if a turn-around request or escape mode request is immediately followed by a stop state instead of the required bridge state.	DPHY	or

1. One field per DPHY RX data lane

2.General MIPICSI2 Packet level interrupt- One interrupt for the packet level errors and status

- Error based interrupt - This interrupt can have a variety of sources as described in the table below :

Table 49-13. General MIPICSI2 Packet level Error based interrupt

Source of error		Details	Error Level	Associated ERRPPST field
Double bit-error		Asserted when an ECC syndrome was computed and two bit-errors are detected in the received header.	Packet	
Single bit-error		Asserted when an ECC syndrome was computed and a single bit-error in the header was detected and corrected. Error Position	Packet	

Table continues on the next page...

Table 49-13. General MIPICSI2 Packet level Error based interrupt (continued)

Source of error		Details	Error Level	Associated ERRPPST field
		(MIPICSI2_ERRPOS) register contains the location where the single bit error was detected, but is only valid if is 1.		
Packet payload error		Asserted when the computed CRC code of the message payload is different than the received CRC code.	Packet	
Frame sync[1]		Asserted when Frame End (FE) is not paired with Frame Starts (FS). Several error cases can be identified for this type of error, such as, when a FS is followed by a second FS the frame corresponding to the first FS is considered an error, or when a packet level double bit-error was signaled from the protocol layer the whole transmission until the first DPHY stop state should be ignored since it contains no information that can be safely decoded and cannot be qualified with a data valid signal.	Protocol Decoding	
Unrecognized ID		Unimplemented or unrecognized ID in the header. Invalid ID Report Register (MIPICSI2_INVID) reports the invalid ID was found and what the invalid ID was.	Protocol Decoding	
Data payload received between FS and FE contains errors		Asserted after FE when the data payload received between FS and FE contains errors. This signal is asserted on any packet payload error and de-asserted on the first FE.	Protocol Decoding	

1. SoT synchronization also generates this error.

- Status based interrupt - This interrupt is triggered on any protocol based status update. After reset all sources are masked. Reading the relevant status register will provide the source. The sources are listed below.
  - Generic short packet received

3.MIPICSI2 data based interrupt. This interrupt is triggered on any error /status based update on the MIPICSI2 data reception

- Error based interrupt - This interrupt can have any of the following sources:

Table 49-14. MIPICSI2 data Error based interrupt

Source of error	Signal	Details	Associated INTRS field
Line Count Error	LCNTE	An error triggered by the hardware if number of lines expected by software are different from the one received by MIPICSI2	LCNT fields in the register Interrupt Status (MIPICSI2_INTRS)
Line Length Error	LLEN	An error triggered by the hardware if length of line expected by software is different from the one received by MIPICSI2	LLEN fields in the register Interrupt Status (MIPICSI2_INTRS)

- Status based interrupt - This interrupt is triggered by any of the following status on MIPICSI2 data reception:

Table 49-15. MIPICSI2 data Status based interrupt

Source of status	Signal	Details	Associated INTRS field
Frame Start	FS	A frame start received by MIPICSI2	FS fields in the register Interrupt Status (MIPICSI2_INTRS)
Frame End	FE	A frame end received by MIPICSI2	FE fields in the register Interrupt Status (MIPICSI2_INTRS)

49.8 Memory map and register definition

NOTE

Only 32-bit access is supported. Non-word accesses do not generate error response and are ignored.

MIPICSI2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	RX Controller Configuration Register (MIPICSI2_CONC)	32	R/W	0000_0000h	49.8.1/2286
4	PHY Configuration Register (MIPICSI2_PHYC)	32	R/W	0000_0002h	49.8.2/2287
8	Clock Configuration Status Register (MIPICSI2_CLKCS)	32	R/W	0000_00C0h	49.8.3/2288
C	D-PHY Lane 0 Configuration Status Register (MIPICSI2_LAN0CS)	32	R/W	0000_0180h	49.8.4/2290
10	D-PHY Data LANE 1 Configuration Status Register (MIPICSI2_LAN1CS)	32	R/W	0000_0180h	49.8.5/2292
14	LANE 2 Configuration/Status Register (MIPICSI2_LAN2CS)	32	R/W	0000_0180h	49.8.6/2294
18	LANE3 Configuration Status Register (MIPICSI2_LAN3CS)	32	R/W	0000_0180h	49.8.7/2296
20	External Resistor Configuration Status Register (MIPICSI2_RESCS)	32	R/W	0000_0000h	49.8.8/2298
28	Status Register (MIPICSI2_SR)	32	R	0000_0000h	49.8.9/2299
2C	DataID Report Register (MIPICSI2_DATAID)	32	R	0000_0000h	49.8.10/ 2300
34	Protocol and Packet Error Register (MIPICSI2_ERRPPREG)	32	w1c	0000_0000h	49.8.11/ 2301
38	Error Position (MIPICSI2_ERRPOS)	32	R	0000_0000h	49.8.12/ 2303
3C	Protocol Packet Error Interrupt Enable (MIPICSI2_ERPPINTEN)	32	R/W	0000_0000h	49.8.13/ 2303
44	PHY Error Report Register (MIPICSI2_ERRPHY)	32	w1c	0000_0000h	49.8.14/ 2305
48	Phy Error Interrupt Enable Register (MIPICSI2_ERPHYIE)	32	R/W	0000_0000h	49.8.15/ 2307
4C	RX Enable Register (MIPICSI2_RXEN)	32	R/W	0000_0000h	49.8.16/ 2310
50	Generic Short Packet Data Register (MIPICSI2_GNSP)	32	R	0000_0000h	49.8.17/ 2310
54	Invalid ID Report Register (MIPICSI2_INVID)	32	R	0000_0000h	49.8.18/ 2311
58	LINE LENGTH (MIPICSI2_LINLEN)	32	R/W	0000_0000h	49.8.19/ 2312
5C	Expected Number of Lines (MIPICSI2_EXPCTDL)	32	R/W	0000_0000h	49.8.20/ 2312
60	Interrupt Enable (MIPICSI2_INTREN)	32	R/W	0000_0000h	49.8.21/ 2313
64	Interrupt Status (MIPICSI2_INTRS)	32	w1c	0000_0000h	49.8.22/ 2314

49.8.1 RX Controller Configuration Register (MIPICSI2_CONC)

This register is used to configure the MIPICSI2 RX Controller.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														NULANE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

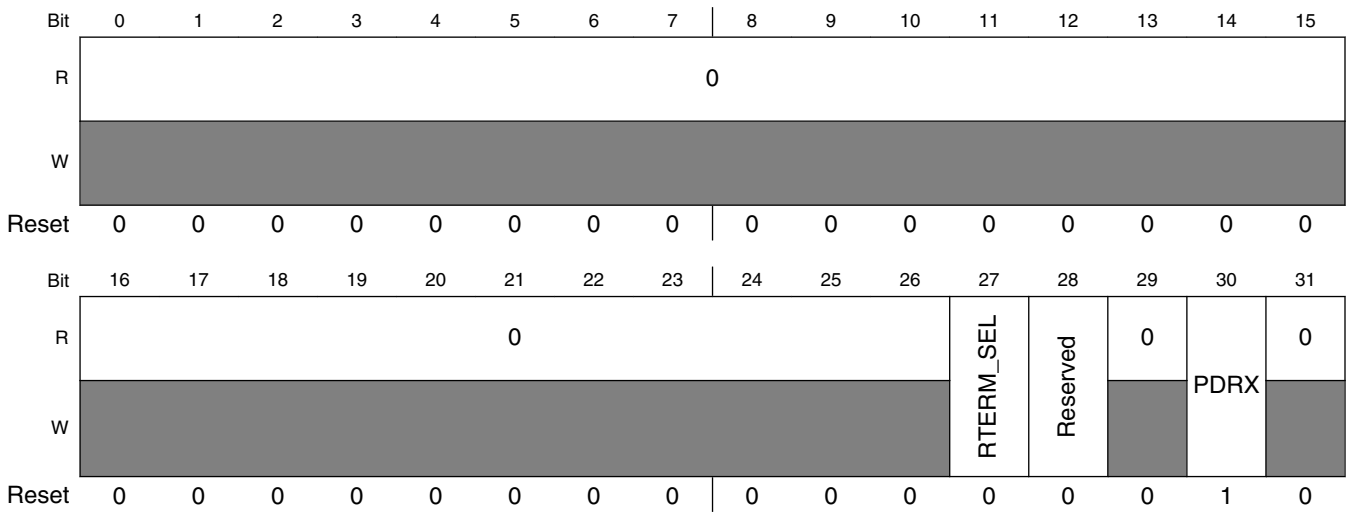
MIPICSI2_CONC field descriptions

Field	Description
0–29 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
30–31 NULANE	Number Of active Lanes being used to receive MIPICSI2 Data This field is to configure the number of data lanes to receive the MIPICSI2 data 00 One Lane to recieve MIPICSI2 data 01 Two Lanes to Receive MIPICSI2 Data 10 Three Lanes to Receive MIPICSI2 Data 11 Four Lanes to Receive MIPICSI2 Data

49.8.2 PHY Configuration Register (MIPICSI2_PHYC)

This register configures the D-PHY.

Address: 0h base + 4h offset = 4h



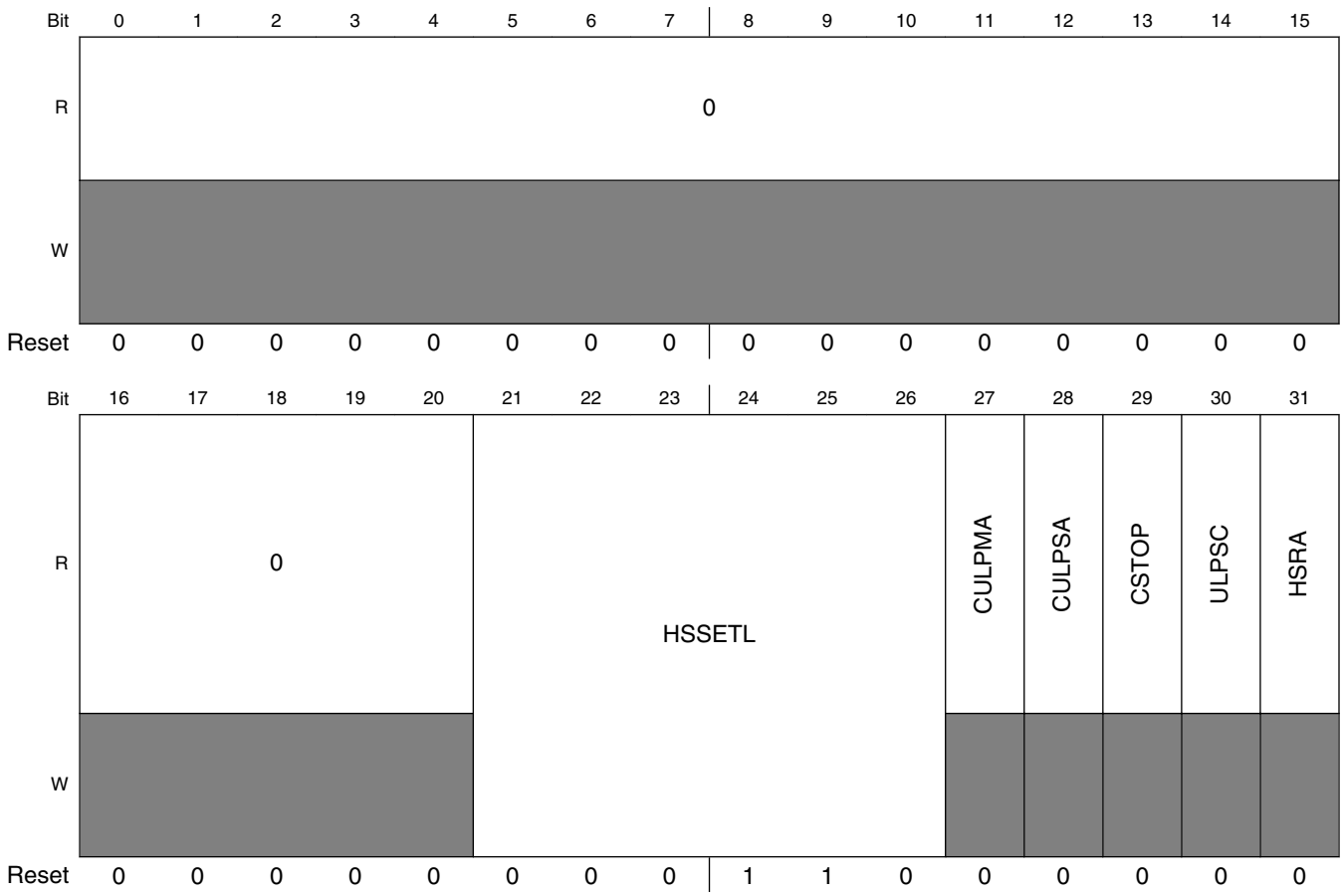
MIPICSI2_PHYC field descriptions

Field	Description
0–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 RTERM_SEL	Voltage Level selection for HS termination This field allows to select voltage level that enables HS termination 0 Low Power Receive level enables High Speed Termination 1 Low Power Contention Detector level enables High Speed Termination
28 Reserved	This field is reserved.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 PDRX	Power Down Receiver This field is the Power Down input for DPHY RX. When high, all blocks are powered down. 0 Power down disabled 1 Power down enabled for all blocks inside PHY
31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

49.8.3 Clock Configuration Status Register (MIPICSI2_CLKCS)

This register is used for configuration and status collection of the clock lane of D-PHY.

Address: 0h base + 8h offset = 8h



MIPICSI2_CLKCS field descriptions

Field	Description
0–20 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
21–26 HSSETL	Clock Lane Program RX HS Settle Time interval during which the HS receiver will ignore any Data Lane HS transitions. Refer to Settle Time for DPHY_RX for more details.
27 CULPMA	Clock Lane ULPS mark Active State This one bit field indicates that lane has exited ULPS and entered Mark-1 state[1]. While in Mark-1 state, High Speed data transmission is not allowed.

Table continues on the next page...

MIPICSI2_CLKCS field descriptions (continued)

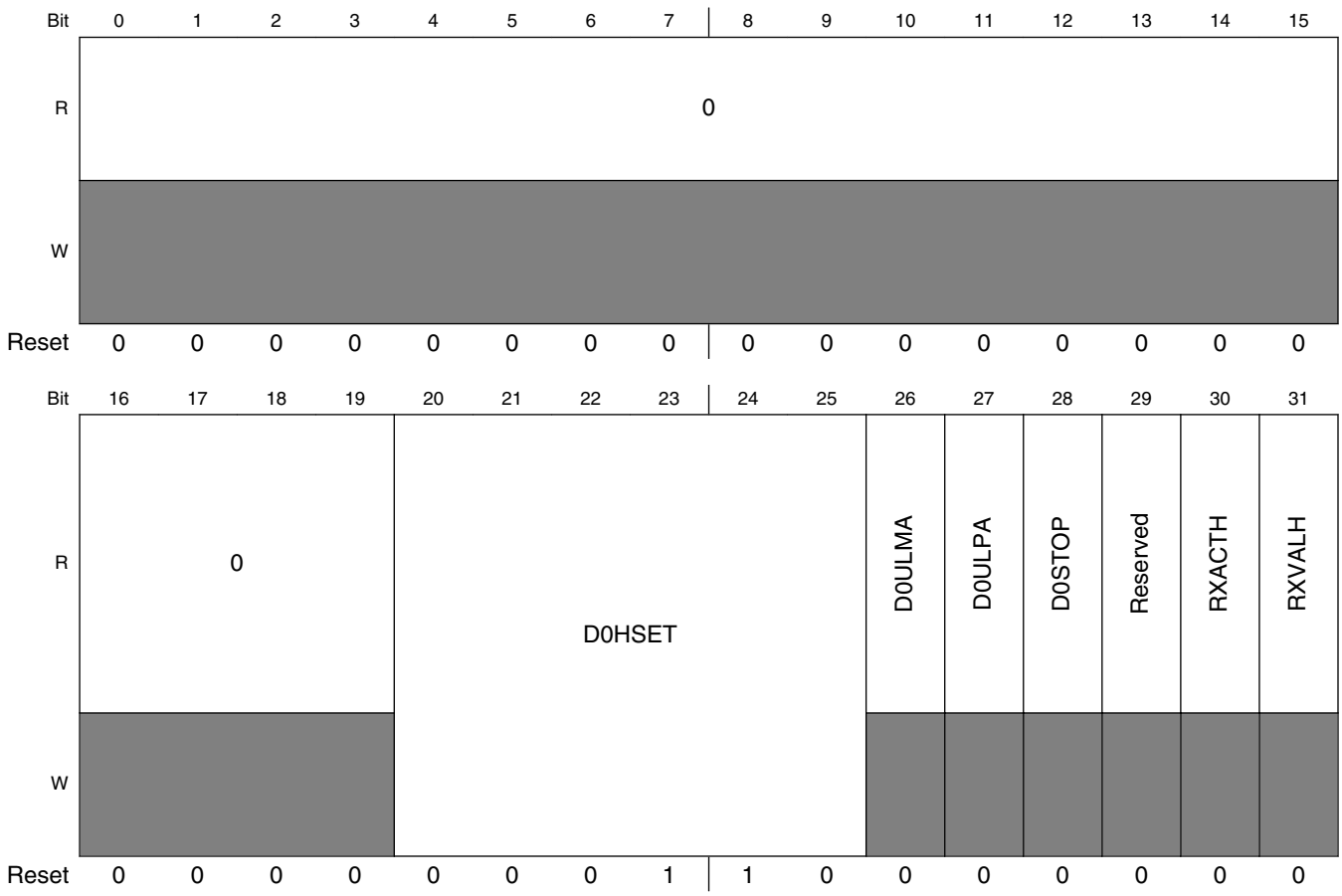
Field	Description
	0 Clock Lane not in Mark active state 1 Clock Lane is in mark active state
28 CULPSA	Clock Lane ULPS Active This one bit field indicates that clock lane is in Ultra Low Power State. This field deasserts the moment the lane enters mark one state. 0 Clock Lane not in ULPS mode 1 Clock Lane in ULPS mode
29 CSTOP	Clock Lane Stop State Clock Lane is in Stop state. This active high signal indicates that the Clock Lane, is currently in Stop state. **NOTE:** Default value would be 0 or 1 depending on when the register is read, before the clock gating is removed or after the clock gating is removed. Refer to Device Clocking Chapter that provides DPHY_ESCAPE_CLK implementation. 0 Clock Lane not in stop state 1 Clock Lane in stop state
30 ULPSC	Clock Lane ULPS This field is asserted to indicate that the Clock Lane Module has entered the Ultra Low-Power State. The Lane Module remains in this mode until a Stop state is detected on the Lane Interconnect. This bit is different from the CULPSA as the CULPSA is deasserted the moment we enter into mark one state, but this field remains asserted after entering the ULPS state till a stop state is detected on the lane. 0 Clock lane not in ultra low power state. 1 Clock Lane in ultra low power state.
31 HSRA	High Speed Clock Receive Active This asynchronous, active high signal indicates that the Clock Lane is receiving a DDR clock signal. 0 DDR clock not being received on the clock lane currently. 1 Clock lane is receiving DDR clock

1. As defined in MIPICSI2 Alliance Specification for D-PHY version 1.1 — 7 November 2011

49.8.4 D-PHY Lane 0 Configuration Status Register (MIPICSI2_LAN0CS)

This register is used for configuring the data lane0 in the D-PHY and for status collection from the lane.

Address: 0h base + Ch offset = Ch



MIPICSI2_LAN0CS field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–25 D0HSET	Data lane 0 Program RX HS Settle time Refer to Settle Time for DPHY_RX for more details.
26 D0ULMA	Data Lane 0 ULPS Mark Active

Table continues on the next page...

MIPICSI2_LAN0CS field descriptions (continued)

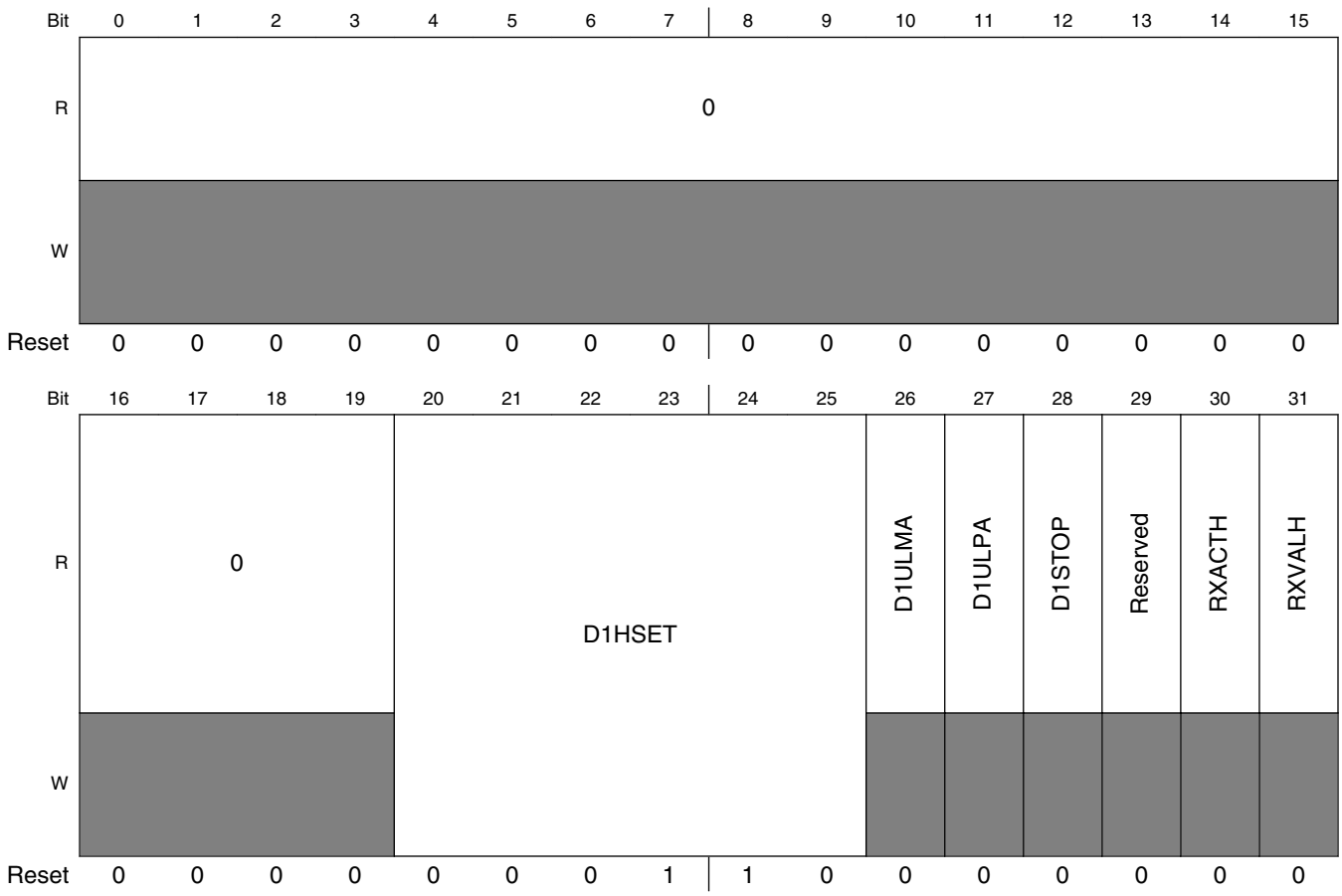
Field	Description
	Receive Ultra Low Power State Mark status. Bit asserts high when data lane 0 has exited ULPS and entered Mark-1 state[1]. While in Mark-1 state, no other active, like High Speed data transmission, is allowed. 0 Data lane 0 is not in Mark 1 state 1 Data lane 0 is in mark 1 state
27 D0ULPA	Data lane 0 ULPS Active Receive Ultra Low Power State active. Bit asserts high when data lane 0 is in ULPS mode. 0 Data Lane 0 ULPS not active 1 Data lane 0 ULPS Active
28 D0STOP	Data Lane 0 Stop State Data Lane 0 is in Stop state. This active high signal indicates that the Data Lane 0, is currently in Stop state. 0 Data lane 0 not in stop state 1 Data lane 0 in stop state.
29 Reserved	This field is reserved.
30 RXACTH	D-PHY Data lane 0 RX active High Speed data This active high field indicates that the lane module is actively receiving high speed data on the lane interconnect. 0 No High Speed data reception ongoing from the lane interconnect. 1 High speed data reception ongoing from lane interconnect.
31 RXVALH	Data Lane 0 RX Valid HS High Speed Receive data on lane 0 is valid. This active high signal indicates that data lane 0 is driving High Speed data on the controller interface. 0 No Valid High Speed data being driven from data lane 0 to MIPICSI2 RX controller 1 Valid High Speed data being driven from data lane 0 to MIPICSI2 RX Controller.

1. As defined in MIPICSI2 Alliance Specification for D-PHY version 1.1 — 7 November 2011

49.8.5 D-PHY Data LANE 1 Configuration Status Register (MIPICSI2_LAN1CS)

This register is used for configuring the Data Lane 1 of D-PHY and collecting the status from this data lane.

Address: 0h base + 10h offset = 10h



MIPICSI2_LAN1CS field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–25 D1HSET	D_PHY Data Lane 1 PRG HS Settle time Time interval during which the data lane 1 HS receiver will ignore any HS transitions on the data lane. Refer to Settle Time for DPHY_RX for more details.
26 D1ULMA	D_PHY Data Lane 1 ULPS Mark active

Table continues on the next page...

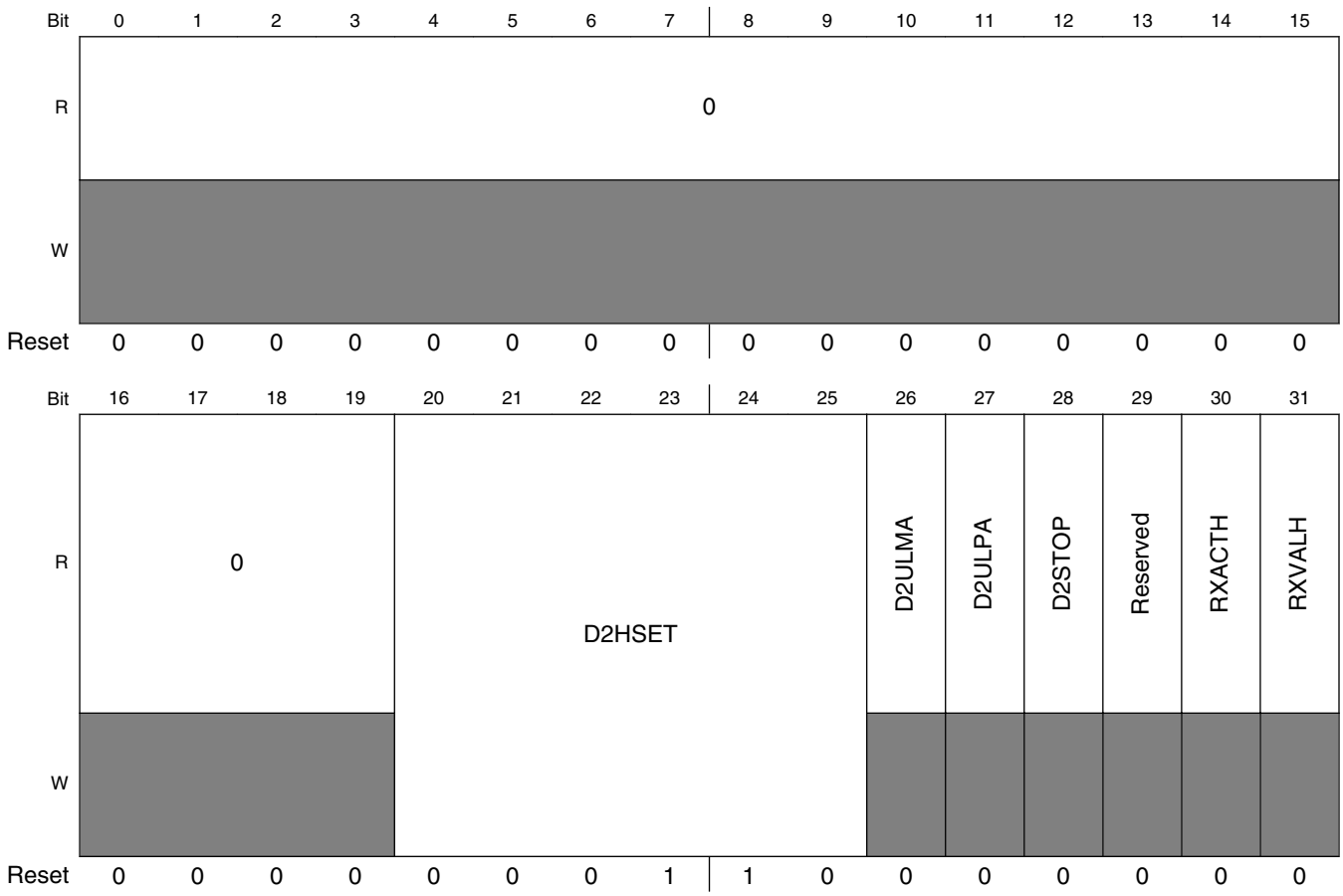
MIPICSI2_LAN1CS field descriptions (continued)

Field	Description
	0 Data Lane 1 not in ULPS mark active state 1 Data Lane 1 in ULPS Mark Active state
27 D1ULPA	D-PHY Data Lane 1 ULPS Active Receive Ultra Low Power State active. Bit asserts high when data lane 1 is in ULPS mode. 0 Data lane not in ULPS state 1 Data Lane 1 in ULPS state
28 D1STOP	D-PHY Data Lane 1 Stop Data Lane 1 is in Stop state. This active high signal indicates that the Data Lane 1 , is currently in Stop state. 0 Data lane 1 is not in stop state 1 Data lane 1 is in stop state
29 Reserved	This field is reserved.
30 RXACTH	D-PHY Data lane 1 Active HS This active high field indicates that the data lane 1 is actively receiving high speed data on the lane interconnect. 0 Data lane 1 not receiving high speed active data on the lane interconnect 1 Data Lane 1 receiving active high speed data on the lane interconnect
31 RXVALH	D-PHY Data lane 1 Receive Valid High speed High Speed Receive data on lane 1 is valid. This active high signal indicates that data lane 1 is driving High Speed data on the controller interface. 0 No Valid High speed data being driven from data lane 1 to the MIPICSI2 RX controller. 1 Valid HS data being transmitted from data lane 1 to the MIPICSI2 RX controller

49.8.6 LANE 2 Configuration/Status Register (MIPICSI2_LAN2CS)

This register is used for configuring the Data Lane 2 of D-PHY and collecting the status from this data lane.

Address: 0h base + 14h offset = 14h



MIPICSI2_LAN2CS field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–25 D2HSET	DPHY Data Lane 2 Sette Time Time interval during which the data lane 2 HS receiver will ignore any HS transitions on the data lane. Refer to Settle Time for DPHY_RX for more details.
26 D2ULMA	DPHY Data Lane 2 Mark One state This field indicates that data lane2 is in mark one state.

Table continues on the next page...

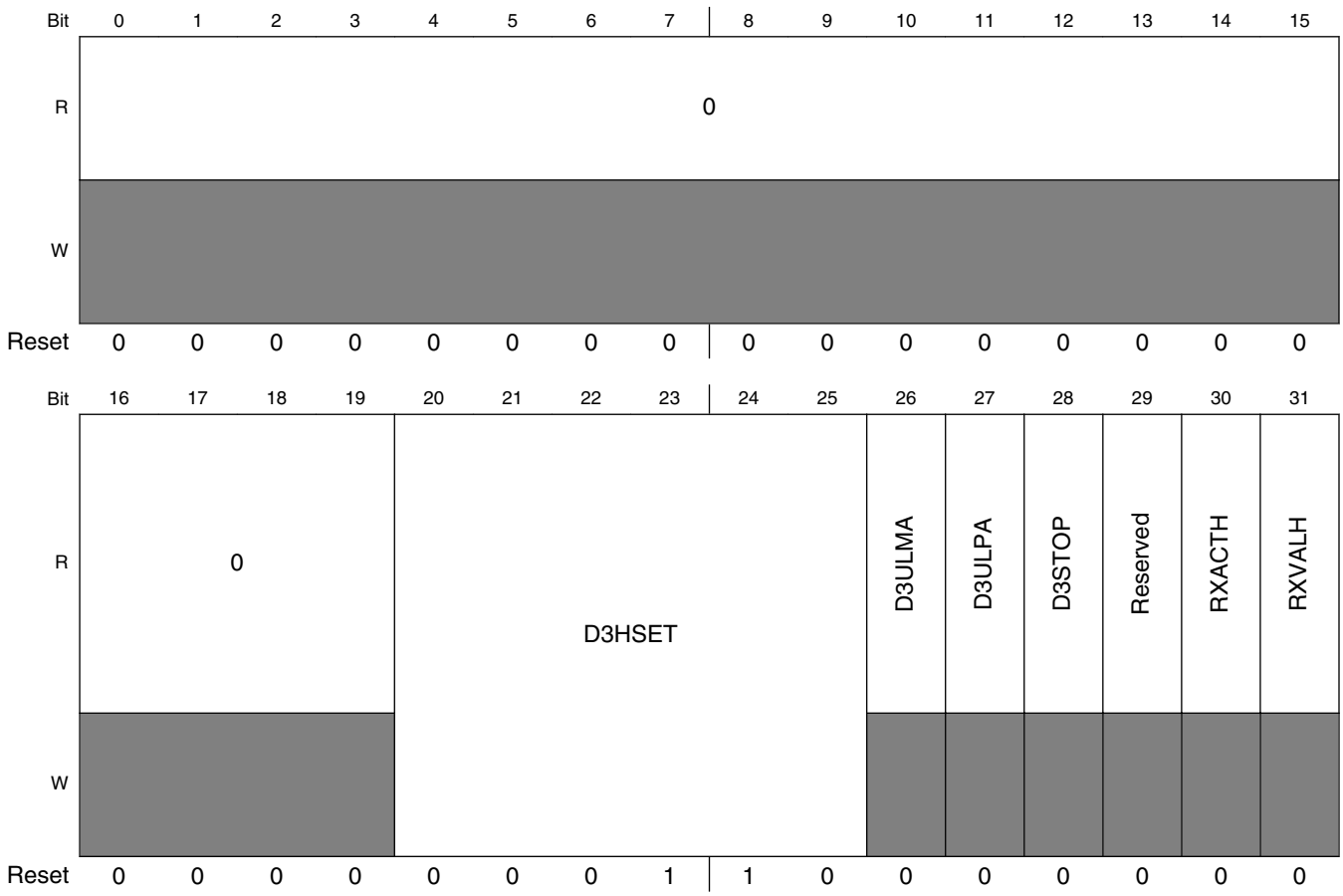
MIPICSI2_LAN2CS field descriptions (continued)

Field	Description
27 D2ULPA	DPHY Dat aLane 2 in Ulps State Receive Ultra Low Power State active. Bit asserts high when data lane 2 is in ULPS mode.
28 D2STOP	DPHY Data Lane 2 Stop State Data Lane 2 is in Stop state. This active high signal indicates that the Data Lane 2 , is currently in Stop state.
29 Reserved	This field is reserved.
30 RXACTH	DPHY Data Lane 2 High Speed Receive Active This active high field indicates that the data lane 2 is actively receiving high speed data on the lane interconnect.
31 RXVALH	High Speed Receive Data Valid High Speed Receive data on lane 2 is vaild. This active high signal indicates that data lane 2 is driving High Speed data on the controller interface.

49.8.7 LANE3 Configuration Status Register (MIPICSI2_LAN3CS)

This register is used for configuring the Data Lane 3 of D-PHY and collecting the status from this data lane.

Address: 0h base + 18h offset = 18h



MIPICSI2_LAN3CS field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–25 D3HSET	DPHY Data Lane 3 High Speed Receive Settle Time Time interval during which the data lane 3 HS receiver will ignore any HS transitions on the data lane. Refer to Settle Time for DPHY_RX for more details.
26 D3ULMA	DPHY Data Lane 3 Mark One State This field indicates that data lane3 is in mark one state.

Table continues on the next page...

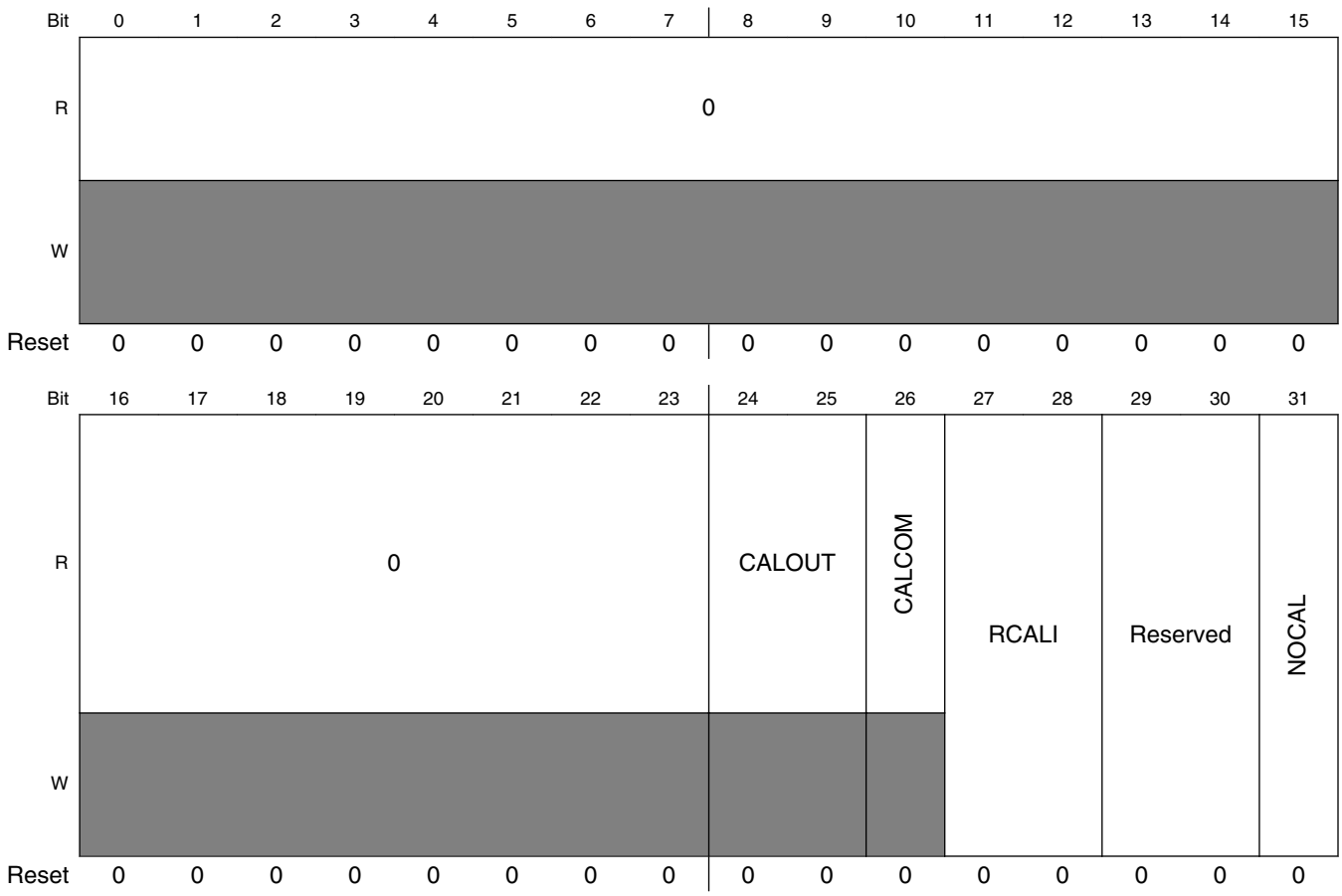
MIPICSI2_LAN3CS field descriptions (continued)

Field	Description
27 D3ULPA	DPHY Data Lane 3 ULPS State Receive Ultra Low Power State active. Bit asserts high when data lane 3 is in ULPS mode.
28 D3STOP	DPHY Data Lane 3 Stop State Data Lane 3 is in Stop state. This active high signal indicates that the Data Lane 3 , is currently in Stop state.
29 Reserved	Reserved This field is reserved. This field is reserved.
30 RXACTH	DPHY Data Lane3 High Speed Receive Active This active high field indicates that the data lane 3 is actively receiving high speed data on the lane interconnect.
31 RXVALH	High Speed Receive Data Vaild High Speed Receive data on lane 3 is vaild. This active high signal indicates that data lane 3 is driving High Speed data on the controller interface.

49.8.8 External Resistor Configuration Status Register (MIPICSI2_RESCS)

This register is used for configuring the external resistor and capturing the status of any calibration done.

Address: 0h base + 20h offset = 20h



MIPICSI2_RESCS field descriptions

Field	Description
0–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
24–25 CALOUT	Calibrator output This field gives the Calibrator Outputs
26 CALCOM	Calibration Complete Used for indicating calibration is complete

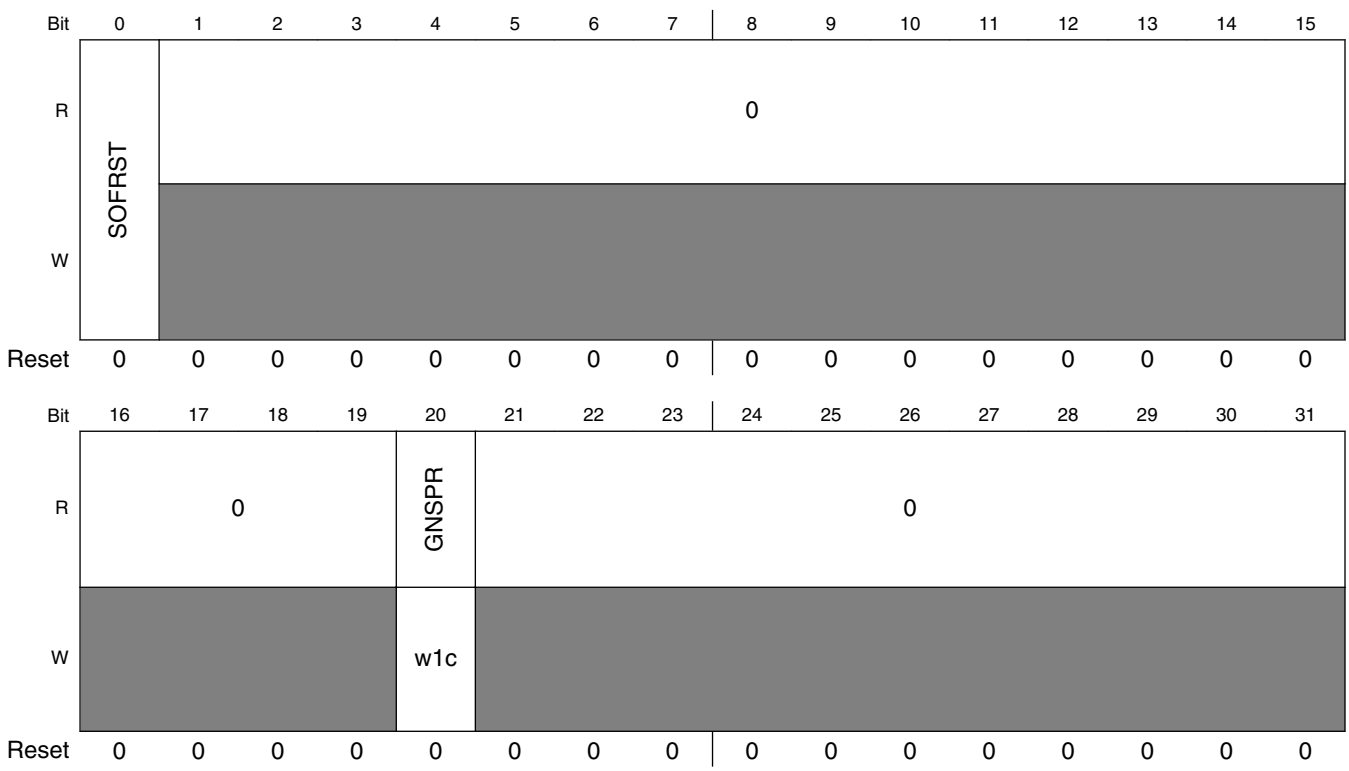
Table continues on the next page...

MIPICSI2_RESCS field descriptions (continued)

Field	Description
	0 Calibration incomplete 1 Calibration Complete
27–28 RCALI	Resistor Calibration Input This field provides On-chip termination control bits for manual calibration. Only active when NOCAL asserted
29–30 Reserved	This field is reserved.
31 NOCAL	No Calibration Used to override calibration value set by auto-calibration circuit with RCALI[1:0]. Set to 0 in auto-calibration operation. 0 Auto Calibration 1 No auto Calibration, manual calibration.

49.8.9 Status Register (MIPICSI2_SR)

Address: 0h base + 28h offset = 28h



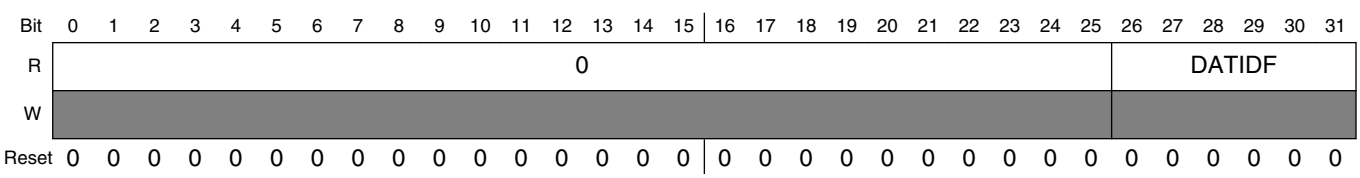
MIPICSI2_SR field descriptions

Field	Description
0 SOFRST	Software reset This field is used to soft reset the IP. The software needs to write a zero to clear this bit. 0 No soft reset requested 1 Soft Reset requested by Software. When set it causes a reset of all the registers and all the FIFOs will be flushed.
1–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20 GNSPR	Generic Short Packet Received This field when set indicates that a generic short packet was received. 0 No generic short packet field was received. 1 Generic Short Packet field was received.
21–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

49.8.10 DataID Report Register (MIPICSI2_DATAID)

This register reports the data type that is being currently received. It is valid only if INVID field in ERRREG is cleared.

Address: 0h base + 2Ch offset = 2Ch



MIPICSI2_DATAID field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26–31 DATIDF	Data ID Field This six bit field reports the current data type being received by the MIPICSI2 subsystem. This field is valid only when INVID field in error register is not set. 0x00h Frame start 0x01h Frame end

Table continues on the next page...

MIPICSI2_DATAID field descriptions (continued)

Field	Description	
	0x02h	Line start
	0x03h	Line end
	0x04h-0X07h	Unused
	0x10h	Null data
	0x11h	Blanking data
	0X12	Embedded data
	0x1Eh	YUV 422 - 8bit data
	0x1Fh	YUV 422 10 bit data
	0x22h	RGB565 data
	0x23h	Unused
	0x24h	RGB888 data
	0x2Ah	RAW8 data
	0x2Bh	RAW10 data
	0x2Ch	RAW12 data
	0x2Dh	RAW14 data
	0x30h-0x37h	User defined data
	0X38h-0X3Fh	Reserved

49.8.11 Protocol and Packet Error Register (MIPICSI2_ERRPPREG)

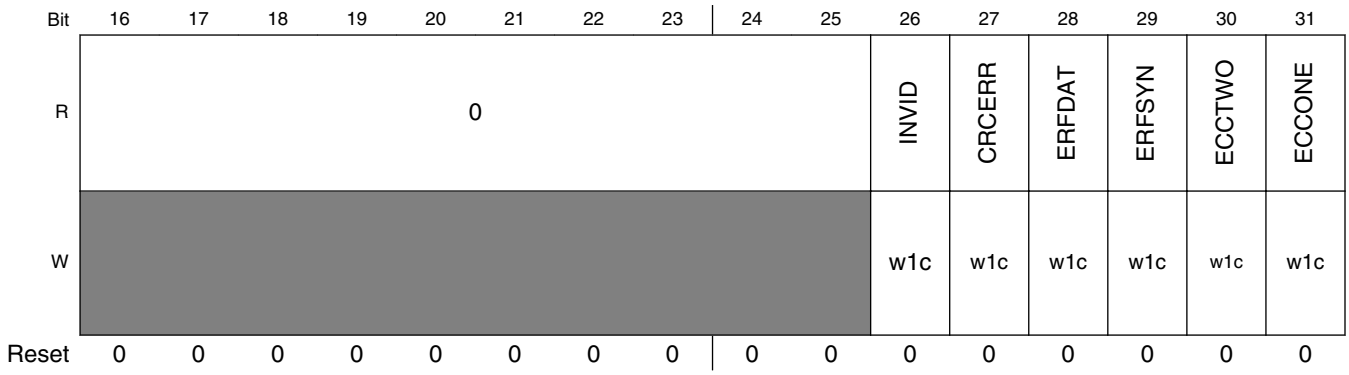
This register stores the errors seen at the receiving side of MIPICSI2 subsystem. It includes protocol and packet level errors.

NOTE

All the bits of this register are W1c , the value of these register fields would indicate: 0 - non occurrence of cause of error or bit cleared 1 - error occurred

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



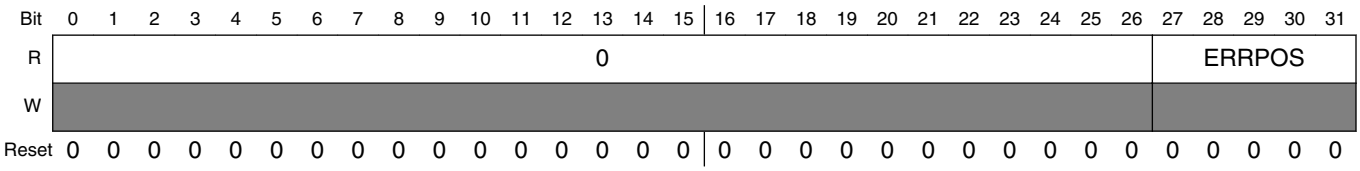
MIPICSI2_ERRPPREG field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 INVID	Invalid ID When set, this field indicates an invalid data type was detected.
27 CRCERR	CRC Error When set , this field indicates a CRC error in the received payload.
28 ERFDAT	Frame data error This field indicates an error in the data within a frame. It sets when a CRC error is detcted on the received payload.
29 ERFSYN	Frame synchronization error When set , this field indicates an error in frame Synchronization. It can be set when : 1.frame start is received after frame start without a frame end in between 2.frame end is received after frame end without a frame start in between 3.Double bit ECC error was detected in received packet header 4.Two bit error in high speed Synchronization pattern was detected.
30 ECCTWO	ECC Two Bit Error When this bit is set it indicates that a two bit ECC error was detected in the received packet header.
31 ECCONE	ECC one bit error When set , this bit indicates that a one bit ECC error was detected and corrected in the received packet header

49.8.12 Error Position (MIPICSI2_ERRPOS)

This register indicates the position at which one bit error was detected in ECC.

Address: 0h base + 38h offset = 38h



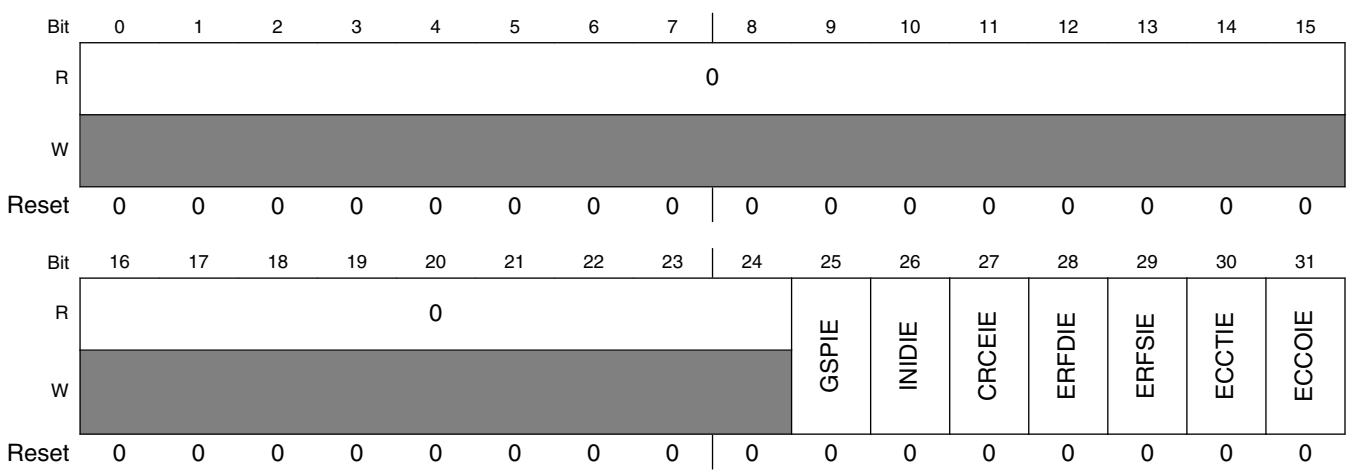
MIPICSI2_ERRPOS field descriptions

Field	Description
0–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27–31 ERRPOS	Error Position This field indicates the location where a one bit error was detected in the ECC by the MIPICSI2 RX controller. The value of this field is valid only if the ECCONE field in the error register is set.

49.8.13 Protocol Packet Error Interrupt Enable (MIPICSI2_ERPPINTEN)

This register is used to enable interrupts corresponding to various errors and status generated at protocol and packet level that are recorded in the error protocol and packet register or status register register.

Address: 0h base + 3Ch offset = 3Ch



MIPICSI2_ERPPINTEN field descriptions

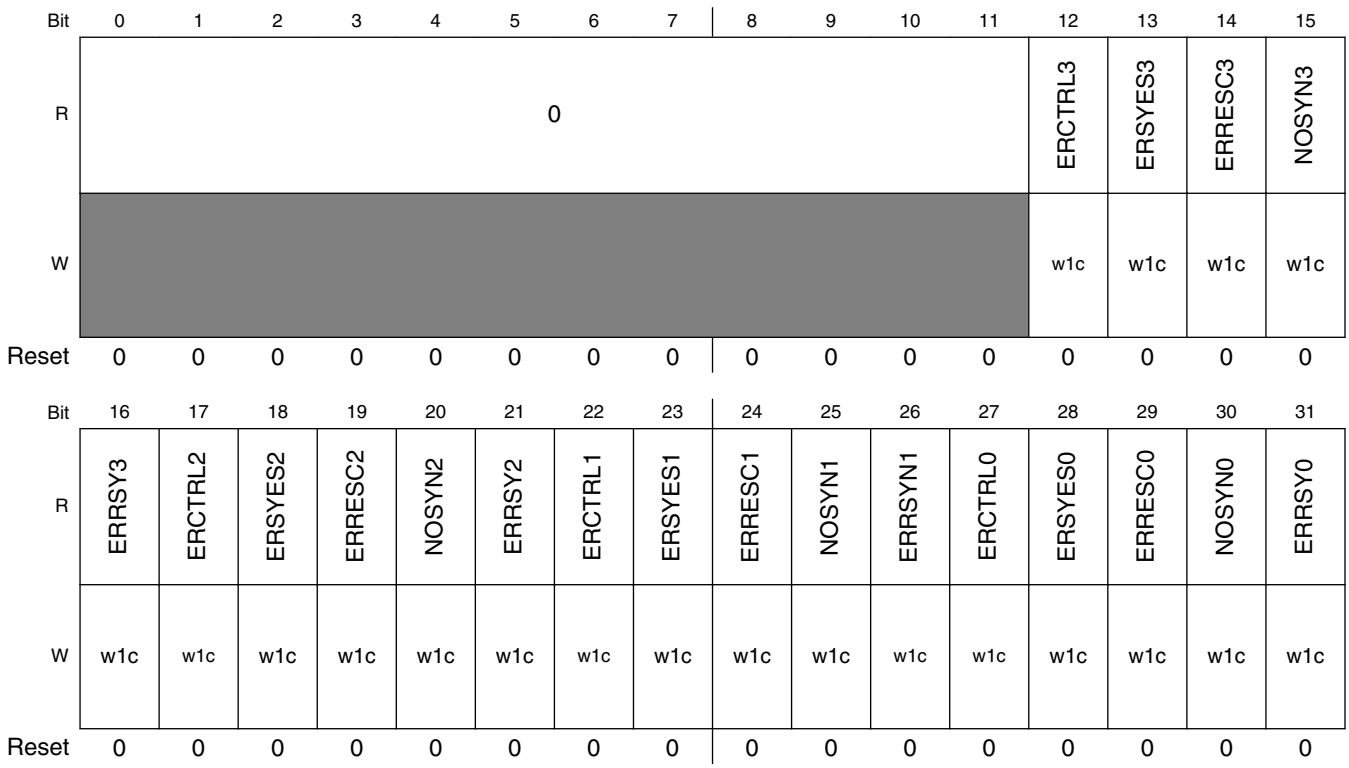
Field	Description
0–24 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
25 GSPIE	Generic Short Packet Interrupt Enable This field enables interrupt generation on reception of a generic short packet . 0 Interrupt on generic short packet reception disabled 1 Interrupt on generic short packet reception enabled
26 INIDIE	Invalid ID Error Interrupt Enable This field enables interrupt generation when the INVID field in the error register is set 0 Interrupt on invalid ID error disabled 1 Interrupt on Invalid ID error enabled
27 CRCEIE	CRC Error Interrupt Enable This field enables interrupt generation when the CRCERR field in the error register is set 0 Interrupt on CRC error disabled 1 Interrupt on CRC error enabled
28 ERFDIE	Error Frame Data Interrupt Enable This field enables interrupt generation when the ERFDAT field in the error register is set 0 Interrupt on frame data error disabled 1 Interrupt on frame data error enabled
29 ERFSIE	Frame Synchronization Error Interrupt Enable This field enables interrupt generation when the ERFSYN field in the error register is set 0 Interrupt on frame Synchronization error disabled 1 Interrupt on frame Synchronization error enabled
30 ECCTIE	ECC Two Interrupt Enable This field enables interrupt generation when the ECCTWO field in the error register is set 0 Interrupt on ECC two bit error disabled 1 Interrupt on ECC two bit error enabled
31 ECCOIE	ECC One Interrupt Enable 0 Interrupt on ECC one bit error disabled 1 Interrupt on ECC one bit error enabled

49.8.14 PHY Error Report Register (MIPICSI2_ERRPHY)

NOTE

All the bits of this register are W1c , the value of these register fields would indicate: 0 - non occurrence of cause of error or bit cleared 1 - error occurred

Address: 0h base + 44h offset = 44h



MIPICSI2_ERRPHY field descriptions

Field	Description
0–11 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
12 ERCTRL3	Control Error on lane 3 When set, this field indicates that an incorrect line state sequence was detected on data lane 3.
13 ERSYES3	Synchronization error in Escape mode on lane 3 When set, this field indicates that a Synchronization error in escape mode on data lane 3 was detected. The field sets if the number of bits received during low power data transmission is not a multiple of eight when the transmission ends.

Table continues on the next page...

MIPICSI2_ERRPHY field descriptions (continued)

Field	Description
14 ERRESC3	Escape Mode Entry Error on lane 3 When set, this field indicates that an error was detected in the escape mode entry sequence on lane 3.
15 NOSYN3	No Synchronization Error Lane 3 When set, this field indicates that more than one bit error was detected in the high speed SoT Synchronization pattern on data lane 3.
16 ERRSY3	Error in the synchronization pattern detected by PHY in lane3 When set , this field indicates that a single bit error in the high speed SOT Synchronization pattern was detected and corrected on data lane 3.
17 ERCTRL2	Control Error on lane 2 When set, this field indicates that an incorrect line state sequence was detected on data lane 2.
18 ERSYES2	Synchronization error in Escape mode on lane 2 When set, this field indicates that a Synchronization error in escape mode on data lane 2 was detected. The field sets if the number of bits received during low power data transmission is not a multiple of eight when the transmission ends.
19 ERRESC2	Escape Mode Entry Error on lane 2 When set, this field indicates that an error was detected in the escape mode entry sequence on lane 2.
20 NOSYN2	No Synchronization Error Lane 2 When set, this field indicates that more than one bit error was detected in the high speed SoT Synchronization pattern on data lane 2.
21 ERRSY2	Error in the synchronization pattern detected by PHY in lane2 When set , this field indicates that a single bit error in the high speed SOT Synchronization pattern was detected and corrected on data lane 2.
22 ERCTRL1	Control Error on lane 1 When set, this field indicates that an incorrect line state sequence was detected on data lane 1.
23 ERSYES1	Synchronization error in Escape mode on lane 1 When set, this field indicates that a Synchronization error in escape mode on data lane 1 was detected. The field sets if the number of bits received during low power data transmission is not a multiple of eight when the transmission ends.
24 ERRESC1	Escape Mode Entry Error on lane 1 When set, this field indicates that an error was detected in the escape mode entry sequence on lane 1.
25 NOSYN1	No Synchronization Error Lane 1 When set, this field indicates that more than one bit error was detected in the high speed SoT Synchronization pattern on data lane 1.
26 ERRSYN1	Error in the synchronization pattern detected by PHY in lane1 When set , this field indicates that a single bit error in the high speed SOT Synchronization pattern was detected and corrected on data lane 1.
27 ERCTRL0	Control Error on lane 0

Table continues on the next page...

MIPICSI2_ERRPHY field descriptions (continued)

Field	Description
	When set, this field indicates that an incorrect line state sequence was detected on data lane 0.
28 ERSYES0	Synchronization error in Escape mode on lane 0 When set, this field indicates that a Synchronization error in escape mode on data lane 0 was detected. The field sets if the number of bits received during low power data transmission is not a multiple of eight when the transmission ends.
29 ERRESC0	Escape Mode Entry Error on lane 0 When set, this field indicates that an error was detected in the escape mode entry sequence on lane 0
30 NOSYN0	No Synchronization Error Lane 0 on HS entry When set, this field indicates that more than one bit error was detected in the high speed SoT Synchronization pattern on data lane 0.
31 ERRSY0	Error in the synchronization pattern detected by PHY in lane0 When set , this field indicates that a single bit error in the high speed SOT Synchronization pattern was detected and corrected on data lane 0.

49.8.15 Phy Error Interrupt Enable Register (MIPICSI2_ERPHYIE)

This register is used to enable interrupts corresponding to various errors and status generated at protocol and packet level that are recorded in the error protocol and packet register or status register.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												ERCLIE3	ERSYIE3	ESERIE3	NOSIE3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERSIE3	ERCLIE2	ERSYIE2	ESERIE2	NOSIE2	ERSIE2	ERCLIE1	ERSYIE1	ESERIE1	NOSIE1	ERSIE1	ERCLIE0	ERSYIE0	ESERIE0	NOSIE0	ERSIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPICSI2_ERPHYIE field descriptions

Field	Description
0–11 Reserved	Reserved This field is reserved.

Table continues on the next page...

MIPICSI2_ERPHYIE field descriptions (continued)

Field	Description
	This read-only field is reserved and always has the value 0.
12 ERCLIE3	This field allows interrupt to be enabled when an incorrect line state sequence transition has been observed. 0 Interrupt disabled on control errors encountered on lane 3 1 Interrupt enabled on control errors encountered on lane 3
13 ERSYIE3	This field is used to enable the interrupt when a synchronization error is observed in the escape mode. 0 interrupt disabled on Synchronization errors in escape mode on lane3 1 interrupt enabled on Synchronization errors in escape mode on lane3
14 ESERIE3	This field is used for enabling interrupt on detection of an escape mode entry error 0 Interrupt disabled on escape mode entry error on lane 3 1 Interrupt enabled on escape mode entry error on lane 3
15 NOSIE3	This field enables interrupt generation when the NOSYN3 field in the error register is set 0 Interrupt disabled on multibit errors in Synchronization error on lane 3 1 Interrupt enabled on multibit errors in Synchronization error on lane 3
16 ERSIE3	This field enables interrupt generation when the ERSYN3 field in the error register is set 0 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 2 disabled 1 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 2 enabled
17 ERCLIE2	This field allows interrupt to be enabled when an incorrect line state sequence transition has been observed. 0 Interrupt disabled on control errors encountered on lane 2 1 Interrupt enabled on control errors encountered on lane 2
18 ERSYIE2	This field is used to enable the interrupt when a synchronization error is observed in the escape mode. 0 interrupt disabled on Synchronization errors in escape mode on lane2 1 interrupt enabled on Synchronization errors in escape mode on lane2
19 ESERIE2	This field is used for enabling interrupt on detection of an escape mode entry error 0 Interrupt disabled on escape mode entry error on lane 2 1 Interrupt enabled on escape mode entry error on lane 2
20 NOSIE2	This field enables interrupt generation when the NOSYN2 field in the error register is set 0 Interrupt disabled on multibit errors in Synchronization error on lane 2 1 Interrupt enabled on multibit errors in Synchronization error on lane 2
21 ERSIE2	This field enables interrupt generation when the ERSYN2 field in the error register is set 0 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 2 disabled 1 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 2 enabled
22 ERCLIE1	This field allows interrupt to be enabled when an incorrect line state sequence transition has been observed. 0 Interrupt disabled on control errors encountered on lane 3 1 Interrupt enabled on control errors encountered on lane 3

Table continues on the next page...

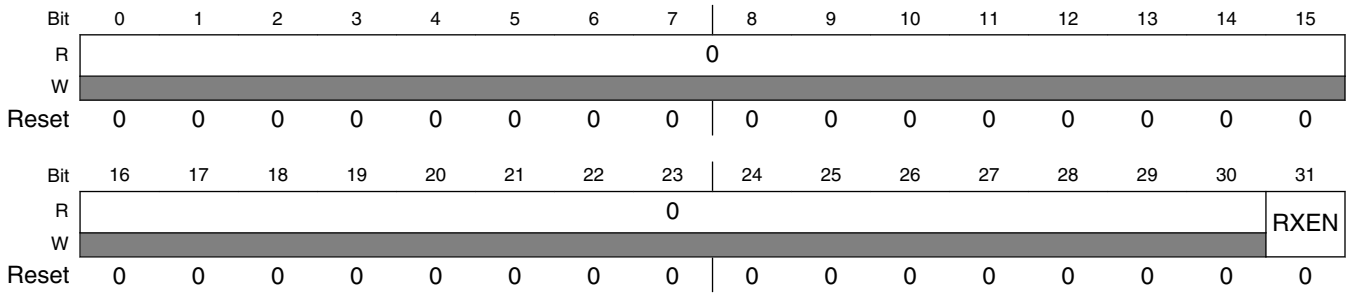
MIPICSI2_ERPHYIE field descriptions (continued)

Field	Description
23 ERSYIE1	This field is used to enable the interrupt when a synchronization error is observed in the escape mode. 0 interrupt disabled on Synchronization errors in escape mode on lane1 1 interrupt enabled on Synchronization errors in escape mode on lane1
24 ESERIE1	This field is used for enabling interrupt on detection of an escape mode entry error 0 Interrupt disabled on escape mode entry error on lane 1 1 Interrupt enabled on escape mode entry error on lane 1
25 NOSIE1	This field enables interrupt generation when the NOSYN1 field in the error register is set 0 Interrupt disabled on multibit errors in Synchronization error on lane 1 1 Interrupt enabled on multibit errors in Synchronization error on lane 1
26 ERSIE1	This field enables interrupt generation when the ERSYN1 field in the error register is set 0 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 1 disabled 1 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 1 enabled
27 ERCLIE0	This field allows interrupt to be enabled when an incorrect line state sequence transition has been observed. 0 Interrupt disabled on control errors encountered on lane 0 1 Interrupt enabled on control errors encountered on lane 0
28 ERSYIE0	This field is used to enable the interrupt when a synchronization error is observed in the escape mode. 0 interrupt disabled on Synchronization errors in escape mode on lane0 1 interrupt enabled on Synchronization errors in escape mode on lane0
29 ESERIE0	This field is used for enabling interrupt on detection of an escape mode entry error 0 Interrupt disabled on escape mode entry error 1 Interrupt enabled on escape mode entry error
30 NOSIE0	This field enables interrupt generation when the NOSYN0 field in the error register is set 0 Interrupt disabled on multibit errors in Synchronization error on lane 0 1 Interrupt enabled on multibit errors in Synchronization error on lane 0
31 ERSIE0	This field enables interrupt generation when the ERSYN0 field in the error register is set 0 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 0 disabled 1 Interrupt on One bit Synchronization error in sync pattern on high speed entry on lane 0 enabled

49.8.16 RX Enable Register (MIPICSI2_RXEN)

This register is used to pause the receive data from the MIPICSI2 controller to the user interface on the next packet boundary.

Address: 0h base + 4Ch offset = 4Ch



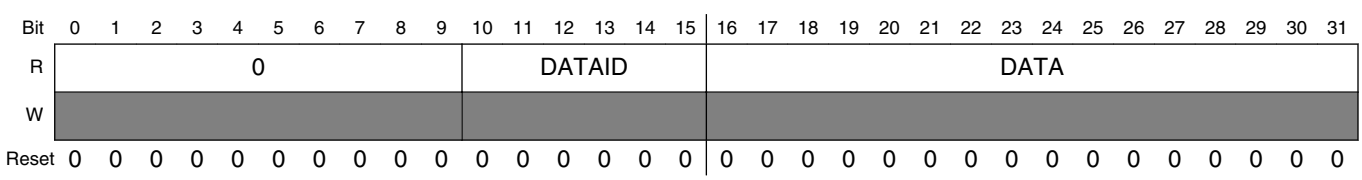
MIPICSI2_RXEN field descriptions

Field	Description
0–30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 RXEN	Receive enable When asserted, this field will enable the receive module for reception of video data from the MIPICSI2 transmitter . When deasserted, the receive module will be disabled . 0 RX disabled 1 RX enabled

49.8.17 Generic Short Packet Data Register (MIPICSI2_GNSP)

This register stores the data field of any generic short packet data received.

Address: 0h base + 50h offset = 50h



MIPICSI2_GNSP field descriptions

Field	Description
0–9 Reserved	Reserved This field is reserved.

Table continues on the next page...

MIPICSI2_GNSP field descriptions (continued)

Field	Description
	This read-only field is reserved and always has the value 0.
10–15 DATAID	Data type This field indicates the type of data in the generic short data packet received. The value of this field is valid only if the field GNSPR in the status register is set. 0x08 Generic short packet code 1 0x09 Generic short packet code 2 0x0A Generic short packet code 3 0x0B Generic short packet code 4 0x0C Generic short packet code 5 0x0D Generic short packet code 6 0x0E Generic short packet code 7 0x0F Generic short packet code 8
16–31 DATA	Data field of a generic short packet This register stores the data field of a generic short packet. The value of this field is valid only if GNSPR field in the status register is set.

49.8.18 Invalid ID Report Register (MIPICSI2_INVID)

This register gives the value of the data id received if the data id is of an unrecognized type.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										INVID					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPICSI2_INVID field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26–31 INVID	Invalid ID This field reports the invalid ID when an unrecognized data type is received. The value of this field is valid only when INVID in error register is set.

49.8.19 LINE LENGTH (MIPICSI2_LINLEN)

This register indicates the length of each line in MIPICSI2 data

NOTE

Number of lines capture should be programmed to a non zero value.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																LINLEN															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPICSI2_LINLEN field descriptions

Field	Description
0–15 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
16–31 LINLEN	LINE LENGTH

49.8.20 Expected Number of Lines (MIPICSI2_EXPCTDL)

This register indicates the expected number of lines in a frame .

NOTE

Total lines expected in a frame should be non zero.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																EXPLINES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPICSI2_EXPCTDL field descriptions

Field	Description
0–15 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

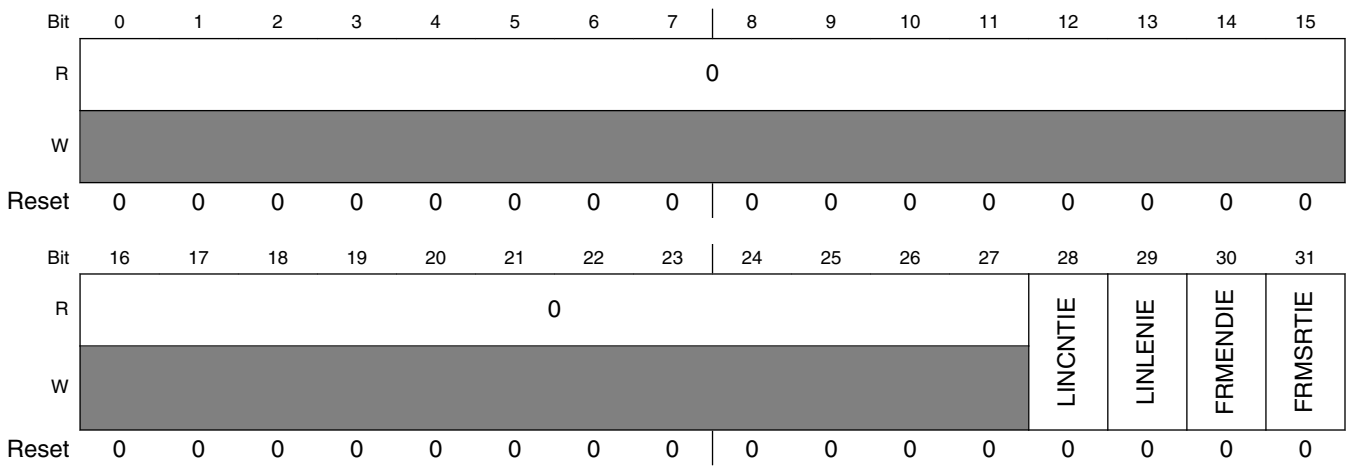
MIPICSI2_EXPCTDL field descriptions (continued)

Field	Description
16–31 EXPLINES	Expected Number Of Lines In a Frame This field allows the programming for the expected number of lines in a frame.

49.8.21 Interrupt Enable (MIPICSI2_INTREN)

This register allows interrupts to be enabled on certain events occurring during MIPICSI2 data capture.

Address: 0h base + 60h offset = 60h



MIPICSI2_INTREN field descriptions

Field	Description
0–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 LINCNTIE	Line Count Error Interrupt Enable This field allows enabling interrupt on a line count error detection. 0 Interrupt on Line Count Error disabled 1 Interrupt on Line Count Error enabled
29 LINLENIE	Line Length Error Interrupt Enable This field allows enabling of interrupt when incorrect line length is observed . 0 Interrupt on Line Length Error Disabled 1 Interrupt on Line Length Error enabled

Table continues on the next page...

MIPICSI2_INTREN field descriptions (continued)

Field	Description
30 FRMENDIE	Frame End Detection Interrupt Enable This field allows enabling interrupt for frame end detection . 0 Interrupt on frame end indication disabled 1 Interrupt on frame end indication enabled
31 FRMSRTIE	Frame Start Detected Interrupt Enable This field allows enabling interrupt when a frame start is detected. 0 Interrupt on frame start indication disabled 1 Interrupt on frame start indication Enabled

49.8.22 Interrupt Status (MIPICSI2_INTRS)

This register indicates the status of interrupt for the events during MIPICSI2 data capture.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LINCNTERR	LINLENERR	FRMEND	FRMSTR
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPICSI2_INTRS field descriptions

Field	Description
0–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 LINCNTERR	Line Count Error Indication This field indicates that an error is detected in the line count from the expected value 0 No Error in line count detected or error cleared 1 Error in line count detected
29 LINLENERR	Line Length Error Indication This field indicates that a line length different from one programmed was received. 0 No error in line length detected or error cleared 1 Error in line length detected
30 FRMEND	Frame End Indication This field indicates that a frame end was detected . 0 Frame End not detected or indication cleared 1 Frame End detected
31 FRMSTR	Frame Start Indication This field indicates that a frame start was detected . 0 Frame Start not detected or indication cleared 1 Frame Start detected

49.9 NOTICE OF DISCLAIMER

The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled by any of the authors or developers of this material or MIPI®. The material contained herein is provided on an “AS IS” basis and to the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and MIPI hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence.

All materials contained herein are protected by copyright laws, and may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of MIPI Alliance.

MIPI, MIPI Alliance and the dotted rainbow arch and all related trademarks, trade names, and other intellectual property are the exclusive property of MIPI Alliance and cannot be used without its express prior written permission.

ALSO, THERE IS NO WARRANTY OF CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT. IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT OR MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT, SPECIFICATION OR DOCUMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Without limiting the generality of this Disclaimer stated above, the user of the contents of this Document is further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this Document; (b) does not monitor or enforce compliance with the contents of this Document; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with the contents of this Document. The use or implementation of the contents of this Document may involve or require the use of intellectual property rights ("IPR") including (but not limited to) patents, patent applications, or copyrights owned by one or more parties, whether or not Members of MIPI. MIPI does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any IPR or claims of IPR as respects the contents of this Document or otherwise.

Chapter 50
System Timer Module (STM)

50.1 Chip specific STM information

50.1.1 System Timer Module (STM) Configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the appropriate system clock divided by an 8-bit prescale value (1 to 256).

This chip includes three STMs, one for each core, with four 32-bit compare channels in each STM.

Table 50-1. System Timer Module (STM) instances

Instance	Description	Number of Channels	Number of Registers (32-bit)
STM_0	Intended for Main Core_0 in the high-speed computational clock domain.	4	14
STM_1	Intended for Main Core_1 in the high-speed computational clock domain.	4	14
STM_2	Intended for Main Core_2 in the high-speed computational clock domain.	4	14

NOTE

An STM interrupt cannot be used for waking up the chip from stop/halt modes.

50.2 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

50.2.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

50.2.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

50.2.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

50.3 External signal description

The STM does not have any external interface signals.

50.4 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

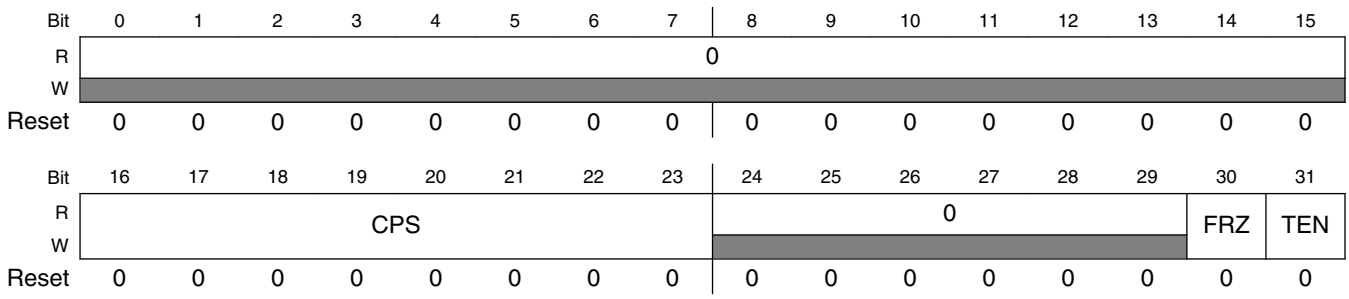
STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STM Control Register (STM_CR)	32	R/W	0000_0000h	50.4.1/2320
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	50.4.2/2321
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	50.4.3/2321
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	50.4.4/2322
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	50.4.5/2322
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	50.4.3/2321
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	50.4.4/2322
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	50.4.5/2322
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	50.4.3/2321
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	50.4.4/2322
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	50.4.5/2322
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	50.4.3/2321
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	50.4.4/2322
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	50.4.5/2322

50.4.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h



STM_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ---------- 0xFF Divide system clock by 256
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **NOTE:** When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

50.4.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

50.4.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CCR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

50.4.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register has the interrupt flag for channel n of the timer.

Address: 0h base + 14h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

50.4.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Address: 0h base + 18h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CMP*n* field descriptions

Field	Description
0–31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

STM_CMP*n* field descriptions (continued)

Field	Description

50.5 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode, the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode; otherwise, it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*), and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

NOTE

The STM counter does not advance when the system clock is stopped.

Chapter 51
Software Watchdog Timer (SWT)

51.1 Chip specific SWT information

51.1.1 Software Watchdog Timer (SWT) Configuration

This device contains three SWTs as shown in the following table:

Table 51-1. SWT Configuration Instances

Instance	Description
SWT_0	Intended for Main Core_0 in the high-speed computational clock domain.
SWT_1	Intended for Main Core_1 in the high-speed computational clock domain.
SWT_2	Intended for Main Core_2 in the high-speed computational clock domain. **NOTE:** SWT2 also has a dependency on CORE_0 core reset. CORE_0 must be out of reset for SWT2 to work. Also by this dependency, CORE_2 cannot be used to execute a code which updates the MC_ME_CADDR1[RMC] or the boot vector for CORE_0. If the z4 core RMC needs to be updated, CORE_1 must be used to do this.

NOTE

Stopping a particular core will not stop SWTs corresponding to other cores. Therefore, these other SWTs may time out if they are enabled. For example, both SWT_1 and SWT_2 are accessible from CORE_0, but these SWTs will not receive stop when CORE_0 is stopped, and may therefore time out.

Each SWT has these features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

NOTE

STP bit in SWT_CR can work for HALT mode also.

51.1.2 Default configuration

NOTE

During sequential boot SWTs would be disabled.

On this device, the SWTs come out of reset with the following default values. Minimum timeout value is 256 cycles for SWT0, SWT1, and SWT2.

Table 51-2. Register reset values for z4 booting core

SWT instance	Register	Reset value
SWT0	SWT0 Control Register (SWT_CR)	See Table 51-3
SWT0	SWT0 Time-out (SWT_TO)	0x0003_FDE0
SWT0	SWT0 Counter Output Register (SWT_CO)	See Table 51-3
SWT1	SWT1 Control Register (SWT_CR)	0xFF00_011A
SWT1	SWT1 Time-out (SWT_TO)	0x0003_FDE0
SWT1	SWT1 Counter Output Register (SWT_CO)	Reset value changes when access to soft lock bit is made before reading CO register
SWT2	SWT2 Control Register (SWT_CR)	0xFF00_011A
SWT2	SWT2 Time-out (SWT_TO)	0x0003_FDE0

Table 51-3. SWT0 Reset values

Enable SWT0 dcl_soc_conf_1 Bit 13	SWT0_CR	SWT0_CO
Enable	0xFF00_011B	0x0000_0000
Disable	0xFF00_011A	Reset value changes when access to soft lock bit is made before reading CO register

Table 51-4. Register reset values for z7 booting core

Register rest values	Register	Reset values
SWT0	SWT0 Control Register (SWT_CR)	0xFF00_011A
SWT0	SWT0 Time-out (SWT_TO)	0x0003_FDE0
SWT0	SWT0 Counter Output Register (SWT_CO)	Reset value changes when access to soft lock bit is made before reading CO register
SWT1	SWT1 Control Register (SWT_CR)	See Table 51-5
SWT1	SWT1 Time-out (SWT_TO)	0x0003_FDE0

Table continues on the next page...

Table 51-4. Register reset values for z7 booting core (continued)

Register rest values	Register	Reset values
SWT1	SWT1 Counter Output Register (SWT_CO)	See Table 51-5
SWT2	SWT2 Control Register (SWT_CR)	0xFF00_011A
SWT2	SWT2 Time-out (SWT_TO)	0x0003_FDE0

Table 51-5. SWT1 Reset values

Enable SWT1 dcl_soc_conf_1 Bit 14	SWT1_CR	SWT1_CO
Enable	0xFF00_011B	0x0000_0000
Disable	0xFF00_011A	Reset value changes when access to soft lock bit is made before reading CO register

The SWT_CR register reset value of 0xFF00_011B selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Soft Locked
- Watchdog is enabled

The SWT_CR register reset value of 0xFF00_011A selects:

- All masters allowed access
- Reset on invalid access
- Oscillator clock selected
- Counter stops in debug mode
- Soft Locked
- Watchdog is disabled

51.1.3 Reset assertion

Each SWT can assert a reset when the watchdog timer expires or when an invalid access is made. To generate the reset, the SWT fault source needs to be configured in the FCCU.

This reset causes a system reset equivalent to assertion of the RESET pin.

51.1.4 Service mode watch point input

Each SWT comes out of reset in 'Fixed Service Sequence' mode (SWT_CR[SMD] = 2’b00), but the Instruction Address Compare 8 (IAC8) register (SPR 568) can be used for servicing by using the SWT jd_watchpt input.

To use SWT address execution modes for this purpose, the e200z4 core IAC8 register signal jd_watchpt[15] is connected to the SWT jd_watchpt input. The jd_watchpt[15] signal tells the SWT that an IAC8 compare occurred.

Set SWT_CR[SMD] to one of the following so the SWT is serviced by executing code at the address loaded into the IAC8 register:

- 2’b10 for Fixed Address Execution; the IAC8 register cannot be updated while the watchdog is enabled. If the SWT is in 'Fixed Address Execution' mode, an SWT output signal tells the core; this ‘freezes’ changes to the IAC8 register when the SWT is enabled.
- 2’b11 for Incremental Address Execution; the IAC8 register can be updated.

51.2 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

51.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out.

51.2.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period

- Programmable selection of window mode or regular servicing
- Programmable selection of reset request or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

51.2.3 Modes of operation

The SWT supports these device modes of operation:

- Normal
- Debug
- Stop

When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode; otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode; otherwise it continues to run.

51.3 External signal description

The SWT module does not have any external interface signals.

51.4 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can be accessed using only 32-bit (word) accesses.

Any of the following attempted accesses are invalid:

- References using non-32-bit access sizes
- Writes to read-only registers

- Writes of incorrect values to SWT_SR when the SWT is enabled
- Accesses to reserved addresses
- Accesses by masters without permission

For an enabled SWT, an invalid access generates a reset request if SWT_CR[RIA] is 1. An invalid access generates a bus error if SWT_CR[RIA] is 0.

If either HLK or SLK in SWT_CR is 1, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SWT Control Register (SWT_CR)	32	R/W	See section	51.4.1/2330
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	51.4.2/2334
8	SWT Time-out Register (SWT_TO)	32	R/W	See section	51.4.3/2334
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	51.4.4/2335
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	51.4.5/2335
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	51.4.6/2336
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	51.4.7/2337

51.4.1 SWT Control Register (SWT_CR)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP0 corresponds to logical IDs 0 and 8. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP1 corresponds to logical IDs 1 and 9. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP2 corresponds to logical IDs 2 and 10. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP3 corresponds to logical IDs 3 and 11. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP4 corresponds to logical IDs 4 and 12. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP5 corresponds to logical IDs 5 and 13.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP6 corresponds to logical IDs 6 and 14. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP7 corresponds to logical IDs 7 and 15. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode 00 Fixed Service Sequence: The watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to SWT_SR. 01 Keyed Service Sequence: The watchdog is serviced by writing two pseudorandom key values to SWT_SR. 10 Fixed Address Execution: The watchdog is serviced by executing code at the address loaded into the designated IAC register, which cannot be updated while the watchdog is enabled. 11 Incremental Address Execution: The watchdog is serviced by executing code at the address loaded into the designated IAC register, which can be updated.
23 RIA	Reset on Invalid Access **NOTE:** For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a reset request if WEN is 1
24 WND	Window Mode 0 Regular mode: service sequence can be done at any time 1 Windowed mode: service sequence is valid only when the down counter is less than the value in the SWT_WN register
25 ITR	Interrupt Then Reset Request

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	NOTE: For a description of how this chip implements SWT reset requests and interrupt requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive time-out
26 HLK	Hard Lock This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

51.4.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

51.4.3 SWT Time-out Register (SWT_TO)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with , when the service sequence is written or when the SWT is enabled.

51.4.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

51.4.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the Soft Lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog; see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the Soft Lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

51.4.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

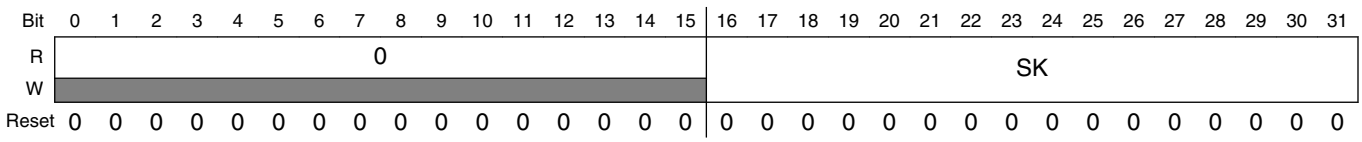
SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN] is 0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN] is 1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to 6 system clock cycles plus 8 counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

51.4.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 18h offset = 18h



SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16} .

51.5 Functional description

51.5.1 Introduction

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),
- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

51.5.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.[3] If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

51.5.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. See the Configuration section to determine the source that is used to clock the down counter.

51.5.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

3. See the chip-specific SWT information.

51.5.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

51.5.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

51.5.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

51.5.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

51.5.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

51.5.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

51.5.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

Figure 51-1. Pseudorandom Key Generator

51.5.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

51.5.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

51.5.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is 1), the service sequence must be applied in the last part of the time-out period defined by SWT_WN.

- The window is open when the down counter is less than the value in SWT_WN.
- Outside this window, service sequence writes are invalid accesses and generate a bus error or reset request depending on the value of SWT_CR[RIA].

For example, if SWT_TO is set to 5000 and SWT_WN is set to 1000, the service sequence must be applied in the last 20% of the time-out period.

Synchronization logic in the watchdog design slightly delays when the window opens. This delay could be up to 3 system clock cycles plus 4 counter clock cycles.

51.5.5 Time-out

The SWT_CR[ITR] bit controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
  - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
  - Clear the interrupt by writing 1 to SWT_IR[TIF].

  If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

51.5.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 52
Periodic Interrupt Timer (PIT)

52.1 Chip-specific PIT information

52.1.1 PIT Instantiations

This device contains two PIT modules each having 4 channels.

52.1.2 PIT/DMA Periodic Trigger Assignments

The PIT generates periodic trigger events to the DMA Mux as shown in the table below. All triggers go to both DMAMUX modules.

Table 52-1. PIT channel assignments for periodic DMA triggering

DMAMUX_0 and DMAMUX_1 Channel Trigger	Source Module	Source Signal
0	PIT_0	Trigger Channel 0
1	PIT_0	Trigger Channel 1
2	PIT_0	Trigger Channel 2
3	PIT_0	Trigger Channel 3
4	PIT_1	Trigger Channel 0
5	PIT_1	Trigger Channel 1
6	PIT_1	Trigger Channel 2
7	PIT_1	Trigger Channel 3

52.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

52.2.1 Block diagram

The following figure shows the block diagram of the PIT module.

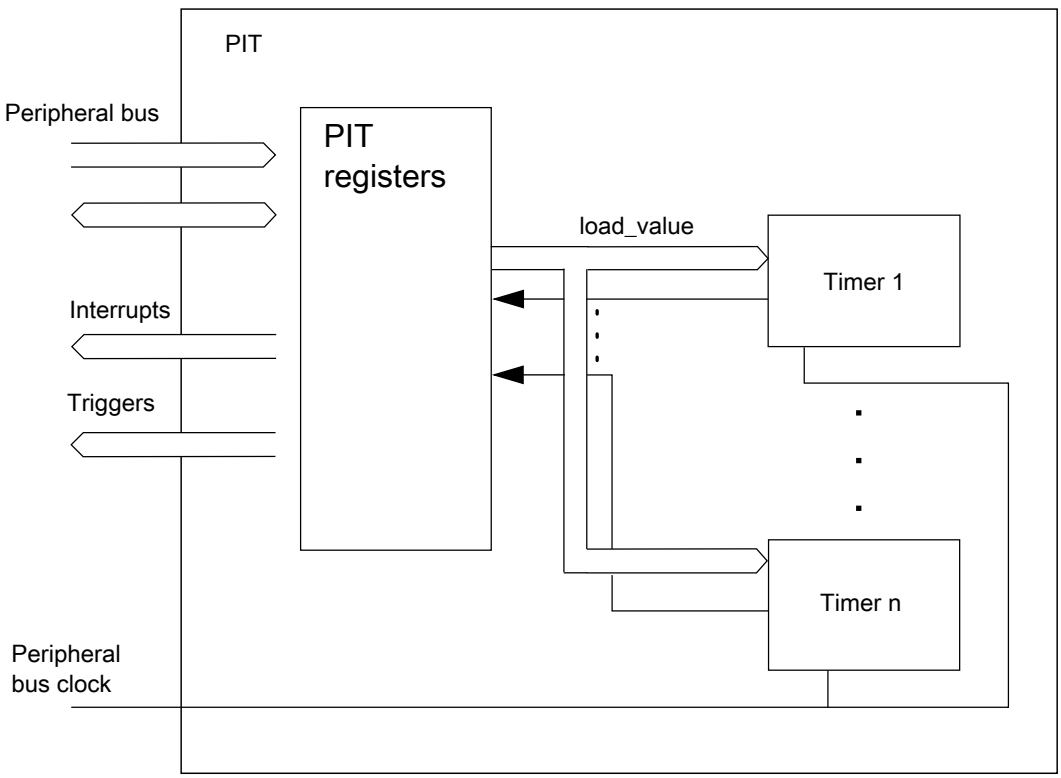


Figure 52-1. Block diagram of the PIT

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

52.2.2 Features

The main features of this block are:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

52.3 Signal description

The PIT module has no external pins.

52.4 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.

PIT memory map

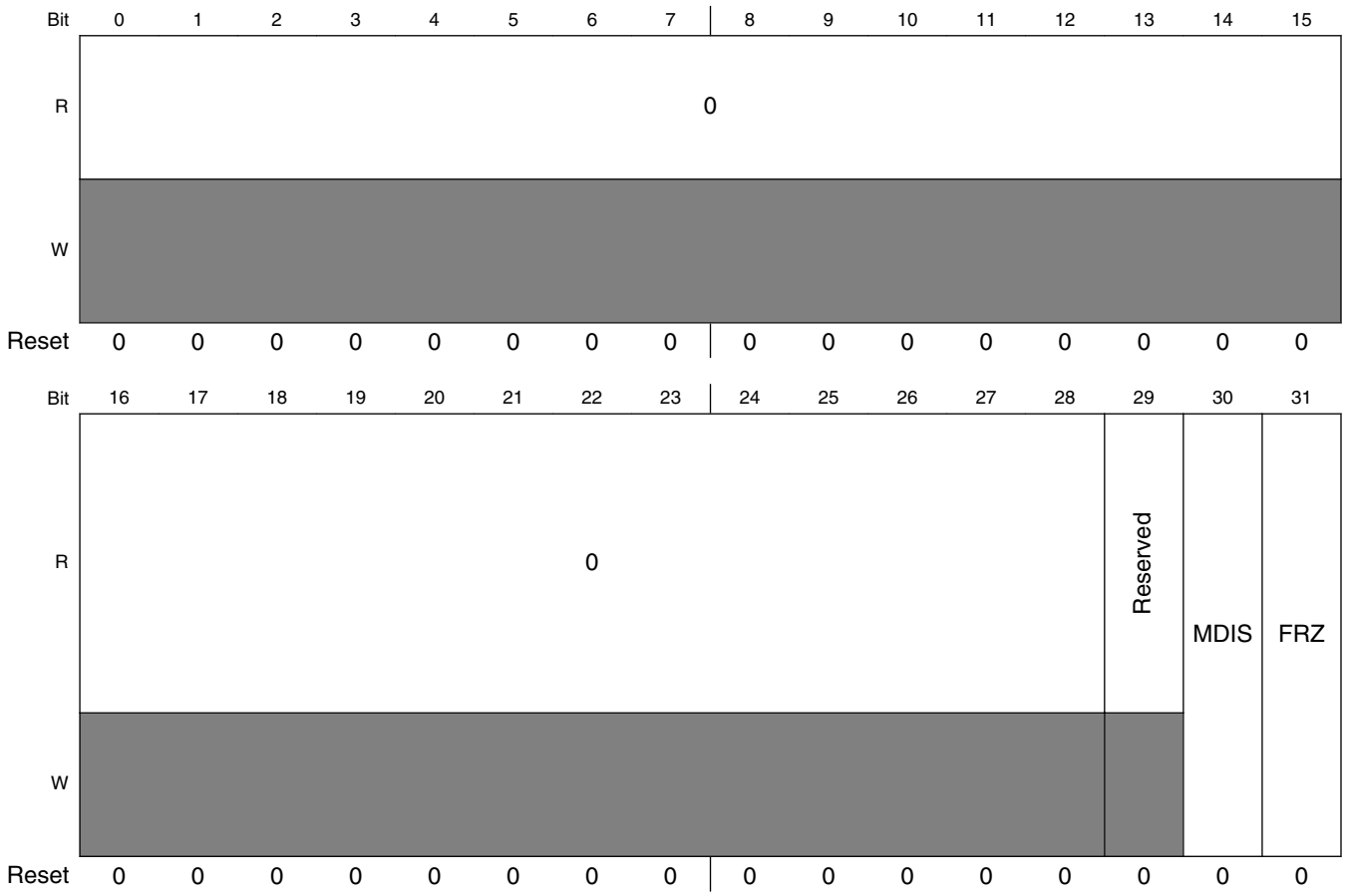
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0000h	52.4.1/2345
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	52.4.2/2347
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	52.4.3/2347
100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	52.4.4/2348
104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	52.4.5/2348
108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	52.4.6/2349
10C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	52.4.7/2350
110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	52.4.4/2348
114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	52.4.5/2348
118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	52.4.6/2349
11C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	52.4.7/2350
120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	52.4.4/2348
124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	52.4.5/2348
128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	52.4.6/2349
12C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	52.4.7/2350
130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	52.4.4/2348
134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	52.4.5/2348
138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	52.4.6/2349
13C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	52.4.7/2350

52.4.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 0h base + 0h offset = 0h



PIT_MCR field descriptions

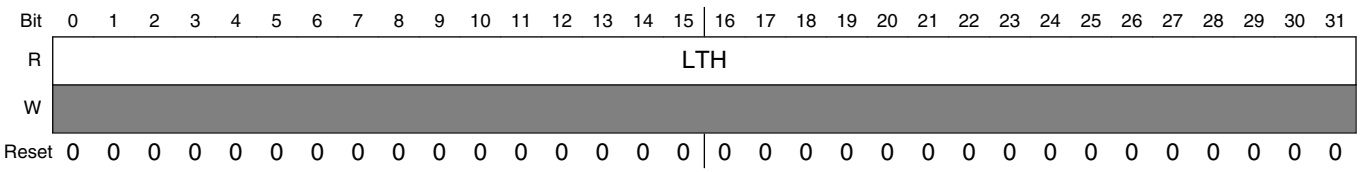
Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved.
30 MDIS	Module Disable - (PIT section) Disables the standard timers. This field must be enabled before any other setup is done. **NOTE:** Always write to this bit after at least 3 bus clock cycles of enabling the PIT clock gate in the device clock generation module. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

52.4.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 0h base + E0h offset = E0h



PIT_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

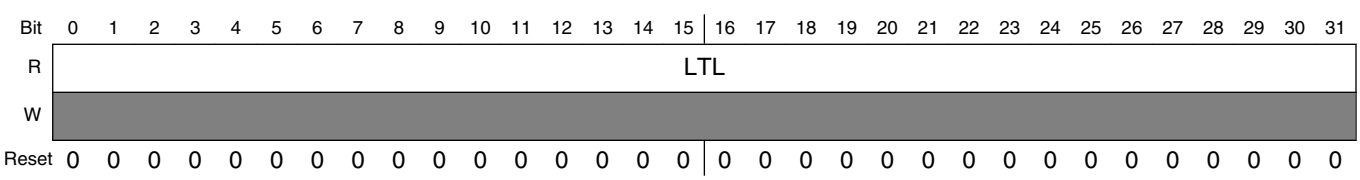
52.4.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 0h base + E4h offset = E4h



PIT_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

52.4.4 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LDVAL*n* field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

52.4.5 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Access: User read only

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL*n* field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled.

PIT_CVAL*n* field descriptions (continued)

Field	Description
	NOTE: • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

52.4.6 Timer Control Register (PIT_TCTRL*n*)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
30 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Enables or disables the timer. 0 Timer n is disabled. 1 Timer n is enabled.

52.4.7 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

52.5 Functional description

This section provides the functional description of the module.

52.5.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

52.5.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

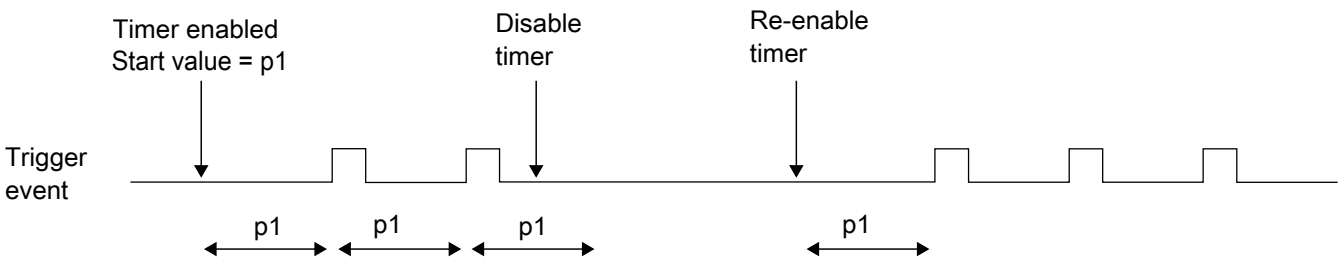


Figure 52-2. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

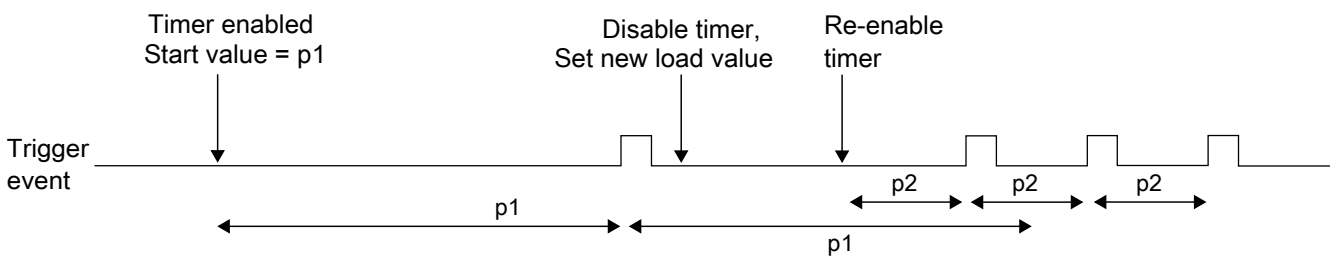


Figure 52-3. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

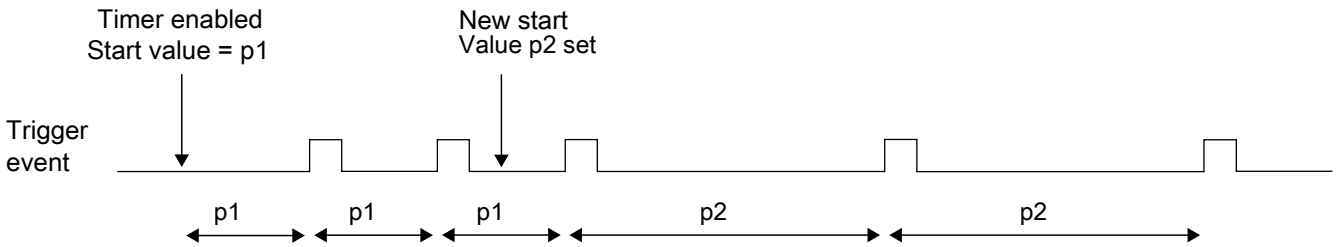


Figure 52-4. Dynamically setting a new load value

52.5.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

52.5.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

52.5.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

52.6 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.

- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

52.7 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

52.8 Example configuration for the lifetime timer

To configure the lifetimer timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1


// Timer 0
```

```
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 53
CAN (FlexCAN)

53.1 Chip-specific FlexCAN information

53.1.1 Overview

The chip contains three individual controller area network (FlexCAN3) modules. All these modules implement the CAN protocol according to Bosch Specification version 2.0B. Additionally, two out of the three modules implement the CAN with flexible data rate (CAN-FD) mode of the Bosch Specification and the ISO/DIS11898-1:2015 with the enhanced CRC.

The CAN-FD protocol capable modules are configured for 96 message buffers. The eDMA request for receive transactions and the pretended networking mode are not supported.

Table 53-1. CAN configuration

Module	CAN mode	CAN-FD mode	Number message buffers	Data rate (Mbps)
CAN_0	Yes	Yes	96	8
CAN_1	Yes	No	64	1
CAN_2	Yes	Yes	96	8

The following table shows the high-level memory map differences between FlexCAN and FlexCAN-FD modules.

Register/bit	CAN_0 and CAN_2 (both support FD)	CAN_1
CAN_MCR[FDEN]	Yes	No
CAN_ECR[RXERRCNT_FAST]	Yes	No

Table continues on the next page...

Register/bit	CAN_0 and CAN_2 (both support FD)	CAN_1
CAN_ECR[TXERRCNT_FAST]		
CAN_ESR1[BIT1ERR_FAST]	Yes	No
CAN_ESR1[BIT0ERR_FAST]	Yes	No
CAN_ESR1[CRCERR_FAST]	Yes	No
CAN_ESR1[FRMERR_FAST]	Yes	No
CAN_ESR1[STFERR_FAST]	Yes	No
CAN_ESR1[ERRINT_FAST]	Yes	No
CAN_ESR1[ERROVR]	Yes	No
CAN_ESR1[BOFFDONEINT]	No	No
CAN_CTRL2[ERRMSK_FAST]	Yes	No
CAN_CTRL2[PREXCEN]	Yes	No
CAN_CTRL2[EDFLTDIS]	Yes	No
CAN_CTRL2[ISOCANFDEN]	Yes	No
CAN_IMASK3	Yes	No
CAN_IFLAG3	Yes	No
CAN_RXIMR[64]-CAN_RXIMR[95]	Yes	No
CAN_FDCTRL	Yes	No
CAN_FDCBT	Yes	No
CAN_FDCRC	Yes	No

There are 3 types of reset mentioned in the FlexCAN module chapter:

- "Chip level hard reset" is any SoC level reset. This could be triggered by assertion of an external reset pin, an LVD/HVD event, software interaction with the Reset Generation Module or any other event that places the device in the reset sequence.
- "Chip level soft reset" is not supported by this device.
- "SOFTRST bit in MCR" is the module level reset control and only resets the FlexCAN module.

53.1.1.1 Device Specific reset values

The reset value of all FlexCAN/FlexCAN FD registers that are device specific are shown in the following table. All other register reset values for FlexCAN/FlexCAN FD are shown in the memory map section.

Table 53-2. FlexCAN reset values

Offset (hex)	Register	Reset value (hex)
34	Control 2 register (CAN1_CTRL2)	0x0080_0000

53.1.1.2 CAN_1 Error Injection Address Register (CAN_ERRIAR) - error injection addresses

The following table lists the error injection addresses given in table for CAN_ERRIAR register is valid for CAN_1 (non-FD).

RAM contents	Injection address	Memory map
FlexCAN Registers	Not mapped	-
MBs	0x0000	0x0080
Reserved	-	0x0480
RXIMRs	0x0400	0x0880
Reserved	-	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	-	0x0A98
RXMGMASK	0x0520	0x0AA0
RXFGMASK	0x0524	0x0AA4
RX14MASK	0x0528	0x0AA8
RX15MASK	0x052C	0x0AAC
SMB_TX[1]	Not mapped	-
SMB_RX0	0x0540	0x0AC0
SMB_RX1	0x0550	0x0AD0
ECC Registers	Not mapped	0x0AE0
Reserved	-	0x0B00

1. If the FlexCAN module supports only classical CAN (CAN 2.0B) i.e. CAN_1, it is not possible to inject errors on TX Serial Message Buffer (TXSMB) address since TXSMB is not implemented on RAM

53.2 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

NOTE

Rx FIFOs cannot be used in FD mode.

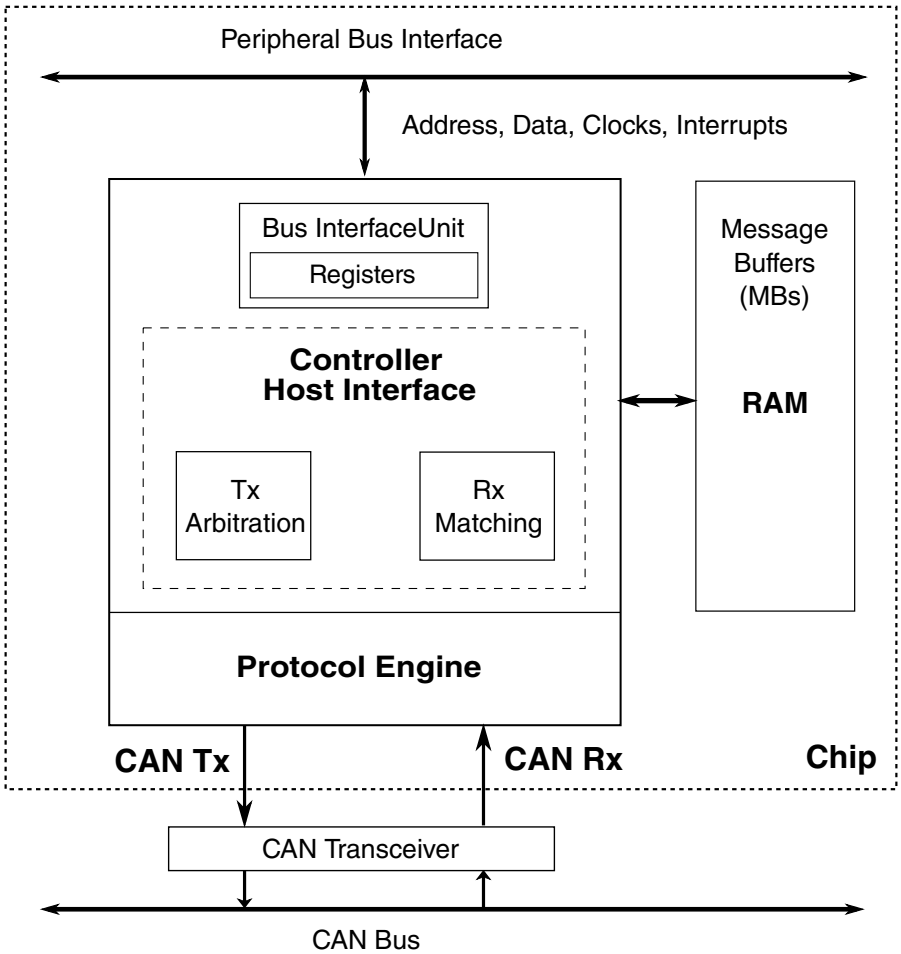


Figure 53-1. FlexCAN block diagram

53.2.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, the CAN with Flexible Data rate (CAN FD) protocol and the CAN 2.0 version B protocol, which supports both standard and extended message frames and long payloads up to 64

bytes transferred at faster rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip configuration details for the actual number of message buffers configured in the chip.

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling
- Detecting CAN FD messages

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the BIU.

53.2.2 FlexCAN module features

The FlexCAN module includes these distinctive features:

- Full implementation of the CAN with Flexible Data Rate (CAN FD) protocol specification and CAN protocol specification, Version 2.0 B
    - Standard data frames
    - Extended data frames
    - Zero to sixty four bytes data length
    - Programmable bit rate (see the chip-specific FlexCAN information for the specific maximum rate configuration)
    - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32 or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox

- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling
- Transmission abort capability
- Flexible message buffers (MBs), totaling 96 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Programmable clock source to the CAN Protocol Interface, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register

- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version
- Supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors)

53.2.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Listen-Only mode:

The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- CAN FD Active mode:

  In this mode, FlexCAN is capable of transmitting and receiving all messages formatted according to the CAN FD Protocol and CAN 2.0 Protocol 2.0 in a interleaved fashion. The CPU can set the FlexCAN into CAN FD Active mode by configuring the MCR[FDEN] bit field in Freeze Mode.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Stop mode:

  This low power mode is entered when Stop mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See Stop mode for more information.

53.3 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 53-3. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

53.3.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

53.3.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

53.4 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

53.4.1 FlexCAN memory mapping

The memory map for the FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 53-4.

Table 53-4. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (CAN_MCR)	S	Yes	Yes
Control 1 register (CAN_CTRL1)	S/U	Yes	No
Free Running Timer register (CAN_TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (CAN_RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (CAN_RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (CAN_RX15MASK)	S/U	No	No

Table continues on the next page...

Table 53-4. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Error Counter Register (CAN_ECR)	S/U	Yes	Yes
Error and Status 1 Register (CAN_ESR1)	S/U	Yes	Yes
Interrupt Masks 2 register (CAN_IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 register (CAN_IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 register (CAN_IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 register (CAN_IFLAG1)	S/U	Yes	Yes
Control 2 Register (CAN_CTRL2)	S/U	Yes	No
Error and Status 2 Register (CAN_ESR2)	S/U	Yes	Yes
CRC Register (CAN_CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (CAN_RXFGMASK)	S/U	No	No
Rx FIFO Information Register (CAN_RXFIR)	S/U	No	No
CAN Bit Timing Register (CAN_CBT)	S/U	Yes	No
Interrupt Masks 3 register (CAN_IMASK3)	S/U	Yes	Yes
Interrupt Flags 3 register (CAN_IFLAG3)	S/U	Yes	Yes
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
Memory Error Control register (CAN_MECR)	S/U	Yes	Yes
Error Injection Address register (CAN_ERRIAR)	S/U	Yes	Yes
Error Injection Data Pattern register (CAN_ERRIDPR)	S/U	Yes	Yes
Error Injection Parity Pattern register (CAN_ERRIPPR)	S/U	Yes	Yes
Error Report Address register (CAN_RERRAR)	S/U	Yes	Yes
Error Report Data register (CAN_RERRDR)	S/U	Yes	Yes
Error Report Syndrome register (CAN_RERRSYNR)	S/U	Yes	Yes
Error Status register (CAN_ERRSR)	S/U	Yes	Yes
CAN FD Control register (CAN_FDCTRL)	S/U	Yes	No
CAN FD Bit Timing register (CAN_FDCBT)	S/U	Yes	No
CAN FD CRC register (CAN_FDCRC)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

The table below shows the FlexCAN memory map.

The address range from offset 0x80 to 0x67F allocates the ninety-six 128-bit Message Buffers (MBs).

The memory maps for the message buffers are in FlexCAN message buffer memory map.

CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (CAN_MCR)	32	R/W	See section	53.4.2/2373
4	Control 1 register (CAN_CTRL1)	32	R/W	0000_0000h	53.4.3/2377
8	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	53.4.4/2381
10	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	Undefined	53.4.5/2382
14	Rx 14 Mask register (CAN_RX14MASK)	32	R/W	Undefined	53.4.6/2383
18	Rx 15 Mask register (CAN_RX15MASK)	32	R/W	Undefined	53.4.7/2384
1C	Error Counter (CAN_ECR)	32	R/W	0000_0000h	53.4.8/2385
20	Error and Status 1 register (CAN_ESR1)	32	R/W	See section	53.4.9/2387
24	Interrupt Masks 2 register (CAN_IMASK2)	32	R/W	0000_0000h	53.4.10/ 2393
28	Interrupt Masks 1 register (CAN_IMASK1)	32	R/W	0000_0000h	53.4.11/ 2394
2C	Interrupt Flags 2 register (CAN_IFLAG2)	32	R/W	0000_0000h	53.4.12/ 2394
30	Interrupt Flags 1 register (CAN_IFLAG1)	32	R/W	0000_0000h	53.4.13/ 2395
34	Control 2 register (CAN_CTRL2)	32	R/W	See section	53.4.14/ 2398
38	Error and Status 2 register (CAN_ESR2)	32	R/W	0000_0000h	53.4.15/ 2401
44	CRC Register (CAN_CRCR)	32	R	0000_0000h	53.4.16/ 2403
48	Rx FIFO Global Mask register (CAN_RXFGMASK)	32	R/W	Undefined	53.4.17/ 2403
4C	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	53.4.18/ 2404
50	CAN Bit Timing Register (CAN_CBT)	32	R/W	See section	53.4.19/ 2405
6C	Interrupt Masks 3 Register (CAN_IMASK3)	32	R/W	0000_0000h	53.4.20/ 2407
74	Interrupt Flags 3 Register (CAN_IFLAG3)	32	R/W	0000_0000h	53.4.21/ 2408
880	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	53.4.22/ 2408
884	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	53.4.22/ 2408
888	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	53.4.22/ 2408
88C	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	53.4.22/ 2408
890	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	53.4.22/ 2408
894	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	53.4.22/ 2408

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
898	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	53.4.22/ 2408
89C	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	53.4.22/ 2408
8A0	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	53.4.22/ 2408
8A4	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	53.4.22/ 2408
8A8	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	53.4.22/ 2408
8AC	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	53.4.22/ 2408
8B0	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	53.4.22/ 2408
8B4	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	53.4.22/ 2408
8B8	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	53.4.22/ 2408
8BC	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	53.4.22/ 2408
8C0	Rx Individual Mask Registers (CAN_RXIMR16)	32	R/W	Undefined	53.4.22/ 2408
8C4	Rx Individual Mask Registers (CAN_RXIMR17)	32	R/W	Undefined	53.4.22/ 2408
8C8	Rx Individual Mask Registers (CAN_RXIMR18)	32	R/W	Undefined	53.4.22/ 2408
8CC	Rx Individual Mask Registers (CAN_RXIMR19)	32	R/W	Undefined	53.4.22/ 2408
8D0	Rx Individual Mask Registers (CAN_RXIMR20)	32	R/W	Undefined	53.4.22/ 2408
8D4	Rx Individual Mask Registers (CAN_RXIMR21)	32	R/W	Undefined	53.4.22/ 2408
8D8	Rx Individual Mask Registers (CAN_RXIMR22)	32	R/W	Undefined	53.4.22/ 2408
8DC	Rx Individual Mask Registers (CAN_RXIMR23)	32	R/W	Undefined	53.4.22/ 2408
8E0	Rx Individual Mask Registers (CAN_RXIMR24)	32	R/W	Undefined	53.4.22/ 2408
8E4	Rx Individual Mask Registers (CAN_RXIMR25)	32	R/W	Undefined	53.4.22/ 2408
8E8	Rx Individual Mask Registers (CAN_RXIMR26)	32	R/W	Undefined	53.4.22/ 2408
8EC	Rx Individual Mask Registers (CAN_RXIMR27)	32	R/W	Undefined	53.4.22/ 2408

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8F0	Rx Individual Mask Registers (CAN_RXIMR28)	32	R/W	Undefined	53.4.22/ 2408
8F4	Rx Individual Mask Registers (CAN_RXIMR29)	32	R/W	Undefined	53.4.22/ 2408
8F8	Rx Individual Mask Registers (CAN_RXIMR30)	32	R/W	Undefined	53.4.22/ 2408
8FC	Rx Individual Mask Registers (CAN_RXIMR31)	32	R/W	Undefined	53.4.22/ 2408
900	Rx Individual Mask Registers (CAN_RXIMR32)	32	R/W	Undefined	53.4.22/ 2408
904	Rx Individual Mask Registers (CAN_RXIMR33)	32	R/W	Undefined	53.4.22/ 2408
908	Rx Individual Mask Registers (CAN_RXIMR34)	32	R/W	Undefined	53.4.22/ 2408
90C	Rx Individual Mask Registers (CAN_RXIMR35)	32	R/W	Undefined	53.4.22/ 2408
910	Rx Individual Mask Registers (CAN_RXIMR36)	32	R/W	Undefined	53.4.22/ 2408
914	Rx Individual Mask Registers (CAN_RXIMR37)	32	R/W	Undefined	53.4.22/ 2408
918	Rx Individual Mask Registers (CAN_RXIMR38)	32	R/W	Undefined	53.4.22/ 2408
91C	Rx Individual Mask Registers (CAN_RXIMR39)	32	R/W	Undefined	53.4.22/ 2408
920	Rx Individual Mask Registers (CAN_RXIMR40)	32	R/W	Undefined	53.4.22/ 2408
924	Rx Individual Mask Registers (CAN_RXIMR41)	32	R/W	Undefined	53.4.22/ 2408
928	Rx Individual Mask Registers (CAN_RXIMR42)	32	R/W	Undefined	53.4.22/ 2408
92C	Rx Individual Mask Registers (CAN_RXIMR43)	32	R/W	Undefined	53.4.22/ 2408
930	Rx Individual Mask Registers (CAN_RXIMR44)	32	R/W	Undefined	53.4.22/ 2408
934	Rx Individual Mask Registers (CAN_RXIMR45)	32	R/W	Undefined	53.4.22/ 2408
938	Rx Individual Mask Registers (CAN_RXIMR46)	32	R/W	Undefined	53.4.22/ 2408
93C	Rx Individual Mask Registers (CAN_RXIMR47)	32	R/W	Undefined	53.4.22/ 2408
940	Rx Individual Mask Registers (CAN_RXIMR48)	32	R/W	Undefined	53.4.22/ 2408
944	Rx Individual Mask Registers (CAN_RXIMR49)	32	R/W	Undefined	53.4.22/ 2408

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
948	Rx Individual Mask Registers (CAN_RXIMR50)	32	R/W	Undefined	53.4.22/ 2408
94C	Rx Individual Mask Registers (CAN_RXIMR51)	32	R/W	Undefined	53.4.22/ 2408
950	Rx Individual Mask Registers (CAN_RXIMR52)	32	R/W	Undefined	53.4.22/ 2408
954	Rx Individual Mask Registers (CAN_RXIMR53)	32	R/W	Undefined	53.4.22/ 2408
958	Rx Individual Mask Registers (CAN_RXIMR54)	32	R/W	Undefined	53.4.22/ 2408
95C	Rx Individual Mask Registers (CAN_RXIMR55)	32	R/W	Undefined	53.4.22/ 2408
960	Rx Individual Mask Registers (CAN_RXIMR56)	32	R/W	Undefined	53.4.22/ 2408
964	Rx Individual Mask Registers (CAN_RXIMR57)	32	R/W	Undefined	53.4.22/ 2408
968	Rx Individual Mask Registers (CAN_RXIMR58)	32	R/W	Undefined	53.4.22/ 2408
96C	Rx Individual Mask Registers (CAN_RXIMR59)	32	R/W	Undefined	53.4.22/ 2408
970	Rx Individual Mask Registers (CAN_RXIMR60)	32	R/W	Undefined	53.4.22/ 2408
974	Rx Individual Mask Registers (CAN_RXIMR61)	32	R/W	Undefined	53.4.22/ 2408
978	Rx Individual Mask Registers (CAN_RXIMR62)	32	R/W	Undefined	53.4.22/ 2408
97C	Rx Individual Mask Registers (CAN_RXIMR63)	32	R/W	Undefined	53.4.22/ 2408
980	Rx Individual Mask Registers (CAN_RXIMR64)	32	R/W	Undefined	53.4.22/ 2408
984	Rx Individual Mask Registers (CAN_RXIMR65)	32	R/W	Undefined	53.4.22/ 2408
988	Rx Individual Mask Registers (CAN_RXIMR66)	32	R/W	Undefined	53.4.22/ 2408
98C	Rx Individual Mask Registers (CAN_RXIMR67)	32	R/W	Undefined	53.4.22/ 2408
990	Rx Individual Mask Registers (CAN_RXIMR68)	32	R/W	Undefined	53.4.22/ 2408
994	Rx Individual Mask Registers (CAN_RXIMR69)	32	R/W	Undefined	53.4.22/ 2408
998	Rx Individual Mask Registers (CAN_RXIMR70)	32	R/W	Undefined	53.4.22/ 2408
99C	Rx Individual Mask Registers (CAN_RXIMR71)	32	R/W	Undefined	53.4.22/ 2408

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9A0	Rx Individual Mask Registers (CAN_RXIMR72)	32	R/W	Undefined	53.4.22/ 2408
9A4	Rx Individual Mask Registers (CAN_RXIMR73)	32	R/W	Undefined	53.4.22/ 2408
9A8	Rx Individual Mask Registers (CAN_RXIMR74)	32	R/W	Undefined	53.4.22/ 2408
9AC	Rx Individual Mask Registers (CAN_RXIMR75)	32	R/W	Undefined	53.4.22/ 2408
9B0	Rx Individual Mask Registers (CAN_RXIMR76)	32	R/W	Undefined	53.4.22/ 2408
9B4	Rx Individual Mask Registers (CAN_RXIMR77)	32	R/W	Undefined	53.4.22/ 2408
9B8	Rx Individual Mask Registers (CAN_RXIMR78)	32	R/W	Undefined	53.4.22/ 2408
9BC	Rx Individual Mask Registers (CAN_RXIMR79)	32	R/W	Undefined	53.4.22/ 2408
9C0	Rx Individual Mask Registers (CAN_RXIMR80)	32	R/W	Undefined	53.4.22/ 2408
9C4	Rx Individual Mask Registers (CAN_RXIMR81)	32	R/W	Undefined	53.4.22/ 2408
9C8	Rx Individual Mask Registers (CAN_RXIMR82)	32	R/W	Undefined	53.4.22/ 2408
9CC	Rx Individual Mask Registers (CAN_RXIMR83)	32	R/W	Undefined	53.4.22/ 2408
9D0	Rx Individual Mask Registers (CAN_RXIMR84)	32	R/W	Undefined	53.4.22/ 2408
9D4	Rx Individual Mask Registers (CAN_RXIMR85)	32	R/W	Undefined	53.4.22/ 2408
9D8	Rx Individual Mask Registers (CAN_RXIMR86)	32	R/W	Undefined	53.4.22/ 2408
9DC	Rx Individual Mask Registers (CAN_RXIMR87)	32	R/W	Undefined	53.4.22/ 2408
9E0	Rx Individual Mask Registers (CAN_RXIMR88)	32	R/W	Undefined	53.4.22/ 2408
9E4	Rx Individual Mask Registers (CAN_RXIMR89)	32	R/W	Undefined	53.4.22/ 2408
9E8	Rx Individual Mask Registers (CAN_RXIMR90)	32	R/W	Undefined	53.4.22/ 2408
9EC	Rx Individual Mask Registers (CAN_RXIMR91)	32	R/W	Undefined	53.4.22/ 2408
9F0	Rx Individual Mask Registers (CAN_RXIMR92)	32	R/W	Undefined	53.4.22/ 2408
9F4	Rx Individual Mask Registers (CAN_RXIMR93)	32	R/W	Undefined	53.4.22/ 2408

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9F8	Rx Individual Mask Registers (CAN_RXIMR94)	32	R/W	Undefined	53.4.22/ 2408
9FC	Rx Individual Mask Registers (CAN_RXIMR95)	32	R/W	Undefined	53.4.22/ 2408
AE0	Memory Error Control Register (CAN_MECR)	32	R/W	800C_0080h	53.4.23/ 2409
AE4	Error Injection Address Register (CAN_ERRIAR)	32	R/W	0000_0000h	53.4.24/ 2411
AE8	Error Injection Data Pattern Register (CAN_ERRIDPR)	32	R/W	0000_0000h	53.4.25/ 2412
AEC	Error Injection Parity Pattern Register (CAN_ERRIPPR)	32	R/W	0000_0000h	53.4.26/ 2413
AF0	Error Report Address Register (CAN_RERRAR)	32	R	0000_0000h	53.4.27/ 2413
AF4	Error Report Data Register (CAN_RERRDR)	32	R	0000_0000h	53.4.28/ 2415
AF8	Error Report Syndrome Register (CAN_RERRSYNR)	32	R	0000_0000h	53.4.29/ 2415
AFC	Error Status Register (CAN_ERRSR)	32	R/W	0000_0000h	53.4.30/ 2418
C00	CAN FD Control Register (CAN_FDCTRL)	32	R/W	See section	53.4.31/ 2419
C04	CAN FD Bit Timing Register (CAN_FDCBT)	32	R/W	See section	53.4.32/ 2423
C08	CAN FD CRC Register (CAN_FDCRC)	32	R	0000_0000h	53.4.33/ 2425

53.4.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FRZ	RFEN	HALT	NOTRDY	Reserved	SOFTRST	FRZACK	SUPV	Reserved	WRNEN	LPMACK	Reserved	Reserved	SRXDIS	IRMQ
W																
Reset	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	LPRIOEN	AEN	FDEN	0	IDAM		0	MAXMB						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CAN_MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This bit is not affected by soft reset. 0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the CAN_MCR Register is set or when Debug mode is requested at chip level. When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. This bit is set by hardware when a non-correctable error is detected and NCEFAFRZ bit in CAN_MECR register is asserted.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze mode. 1 Enabled to enter Freeze mode.
2 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CAN_CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing). This bit can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** This bit cannot be set when CAN FD operation is enabled (see FDEN bit). 0 Rx FIFO not enabled. 1 Rx FIFO enabled.
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and the Control Registers CAN_CTRL1 and CAN_CTRL2. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode. The HALT bit is set by hardware when a non-correctable error is detected and NCEFAFRZ bit in CAN_MECR register is asserted. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
4 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode, Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes. This bit is not affected by soft reset. 0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1 FlexCAN module is either in Disable mode, Stop mode or Freeze mode.
5 Reserved	This field is reserved. When writing to this field, always write the reset value.
6 SOFTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at chip level. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. This bit is not affected by soft reset. 0 No reset request. 1 Resets the registers affected by soft reset.
7 FRZACK	Freeze Mode Acknowledge

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". This bit is not affected by soft reset. **NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses. 1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
9 Reserved	This field is reserved. When writing to this field, always write the reset value.
10 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
11 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. This bit is not affected by soft reset. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing"). 0 FlexCAN is not in a low-power mode. 1 FlexCAN is in a low-power mode.
12 Reserved	This field is reserved. When writing to this field, always write the reset value.
13 Reserved	This field is reserved. When writing to this field, always write the reset value.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
14 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Self reception enabled. 1 Self reception disabled.
15 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK and CAN_RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
19 AEN	Abort Enable When asserted, this bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When CAN_MCR[AEN] is asserted, only the abort mechanism (see Transmission abort mechanism) must be used for updating Mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the CAN_MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.
20 FDEN	CAN FD operation enable This bit enables the CAN with Flexible Data rate (CAN FD) operation. This bit can be written in Freeze mode only. **NOTE:** The Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	1 CAN FD is enabled. FlexCAN is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. 0 CAN FD is disabled. FlexCAN is able to receive and transmit messages in CAN 2.0 format.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than or equal to the number of available Message Buffers, as described in FlexCAN Memory Partition for CAN FD. Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by RFFN bit in CAN_CTRL2 register. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing).

53.4.3 Control 1 register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW) can also be configured in CAN_CBT register, which extends the range of all these variables. If CAN_CBT[BTF] is set, PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW fields of CAN_CTRL1 become read only.

NOTE

When the CAN FD feature is enabled, do not use the PRESDIV, RJW, PSEG1, PSEG2, and PROPSEG fields of the CAN_CTRL1 register for CAN bit timing. Instead use the the CAN_CBT register's EPRESDIV, ERJW, EPSEG1, EPSEG2, and EPROPSEG fields.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL1 field descriptions

Field	Description
0–7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
10–12 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
13–15 PSEG2	Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
16 BOFFMSK	Bus Off Interrupt Mask This bit provides a mask for the Bus Off Interrupt BOFFINT in CAN_ESR1 register. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERRMSK	Error Interrupt Mask This bit provides a mask for the Error Interrupt ERRINT in the CAN_ESR1 register. 0 Error interrupt disabled. 1 Error interrupt enabled.
18 CLKSRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock or the oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Protocol timing". 0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock. 1 The CAN engine clock source is the peripheral clock.
19 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, the CAN_MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	NOTE: The TDCEN bit in CAN_FDCTRL register must be disabled when LPB is asserted. 0 Loop Back disabled. 1 Loop Back enabled.
20 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRNMSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
22 Reserved	This field is reserved.
23 Reserved	This field is reserved.
24 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** For proper operation, to assert SMP it is necessary to guarantee a minimum value of 2 TQs in CAN_CTRL1[PSEG1] (or CAN_CBT[EPSEG1]).This bit cannot be asserted when CAN FD is enabled (CAN_MCR[FDEN] = 1). 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
25 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special "SYNC" message, that is, global network time. If the RFEN bit in CAN_MCR is set (Rx FIFO enabled), the first available Mailbox, according to CAN_CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the CAN_MCR[LPRIOEN] bit does not affect the priority arbitration. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in CAN_ECR register are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (RXERRCNT) in CAN_ECR register, as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of CAN_ESR1[FLTCONF] field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Listen-Only mode is deactivated. 1 FlexCAN module operates in Listen-Only mode.
29–31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

53.4.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Stop and Freeze modes.

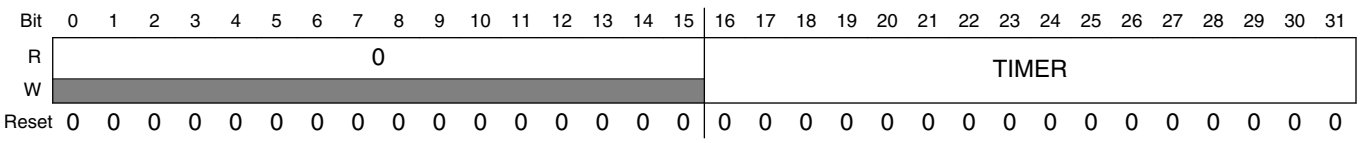
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CAN_CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CAN_CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure, see Section "Mailbox Lock Mechanism".

Address: 0h base + 8h offset = 8h



CAN_TIMER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TIMER	Timer Value Contains the free-running counter value.

53.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the CAN_MCR[IRMQ] bit is negated, RXMGMASK is always in effect (the bits in the MG field will mask the Mailbox filter bits).
- When the CAN_MCR[IRMQ] bit is asserted, RXMGMASK has no effect (the bits in the MG field will not mask the Mailbox filter bits).

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXMGMASK field descriptions

Field	Description
0–31 MG	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

SMB[RTR] [1]	CAN_CTRL2[RRS]	CAN_CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	note [2]	note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

53.4.6 Rx 14 Mask register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX14M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX14MASK field descriptions

Field	Description
0–31 RX14M	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

53.4.7 Rx 15 Mask register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX15M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX15MASK field descriptions

Field	Description
0–31 RX15M	Rx Buffer 15 Mask Bits

CAN_RX15MASK field descriptions (continued)

Field	Description
	Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

53.4.8 Error Counter (CAN_ECR)

This register has four 8-bit fields reflecting the value of the FlexCAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)
- Transmit Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (TXERRCNT_FAST field)
- Receive Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (RXERRCNT_FAST field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST are dedicated to count only the errors occurred in the Data Phase of CAN FD frames with the BRS bit set.

The Fault Confinement State (FLTCONF field in Error and Status Register 1 - CAN_ESR1) is updated based on TXERRCNT and RXERRCNT counters only. TXERRCNT and RXERRCNT counters can be written in Freeze mode only. TXERRCNT_FAST and RXERRCNT_FAST counters are read-only except in Freeze mode where the CPU can write value zero. The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect "Error Passive" state.
- If the FlexCAN state is "Error Passive", and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect "Error Active" state.

- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect "Bus Off" state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in "Bus Off' state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be "Error Active" and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during busoff.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to "Error Passive" state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the "Bus Off" state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to "Error Active" state.
- TXERRCNT_FAST and RXERRCNT_FAST error counters values increment and decrement based on errors detected only in the Data Phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen while FlexCAN is in "Bus Off" state. They are reset when FlexCAN leaves "Bus Off" state and restart counting once FlexCAN resumes to "Error Active" state.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RXERRCNT_FAST								TXERRCNT_FAST								RXERRCNT								TXERRCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ECR field descriptions

Field	Description
0–7 RXERRCNT_ FAST	Receive Error Counter for fast bits Receive Error Counter for errors detected in the Data Phase of received CAN FD messages with the BRS bit set. The RXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write a 8-bit zero value only.

Table continues on the next page...

CAN_ECR field descriptions (continued)

Field	Description
8–15 TXERRCNT_ FAST	Transmit Error Counter for fast bits Transmit Error Counter for errors detected in the Data Phase of transmitted CAN FD messages with the BRS bit set. The TXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write a 8-bit zero value only.
16–23 RXERRCNT	Receive Error Counter Receive Error Counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.
24–31 TXERRCNT	Transmit Error Counter Transmit Error Counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.

53.4.9 Error and Status 1 register (CAN_ESR1)

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device and it is the source of some interrupts to the CPU.

The reported error conditions are BIT1ERR, BIT0ERR, ACKKERR, CRCERR, FRMERR and STFERR, for errors detected in CAN frames of any format, and BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST and STFERR_FAST for errors detected in the Data Phase of CAN FD frames with the BRS bit set only.

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cummulative in case more error events happen in the next frames while the CPU does not attempt to read this register.

TXWRN, RXWRN, IDLE, TX, FLTCONF, RX and SYNCH are status bits.

BOFFINT, BOFFDONEINT, ERRINT, ERRINT_FAST, TWRNINT and RWRNINT are interrupt bits. It is recommended the CPU to use the following procedure when servicing interrupt requests generated by these bits:

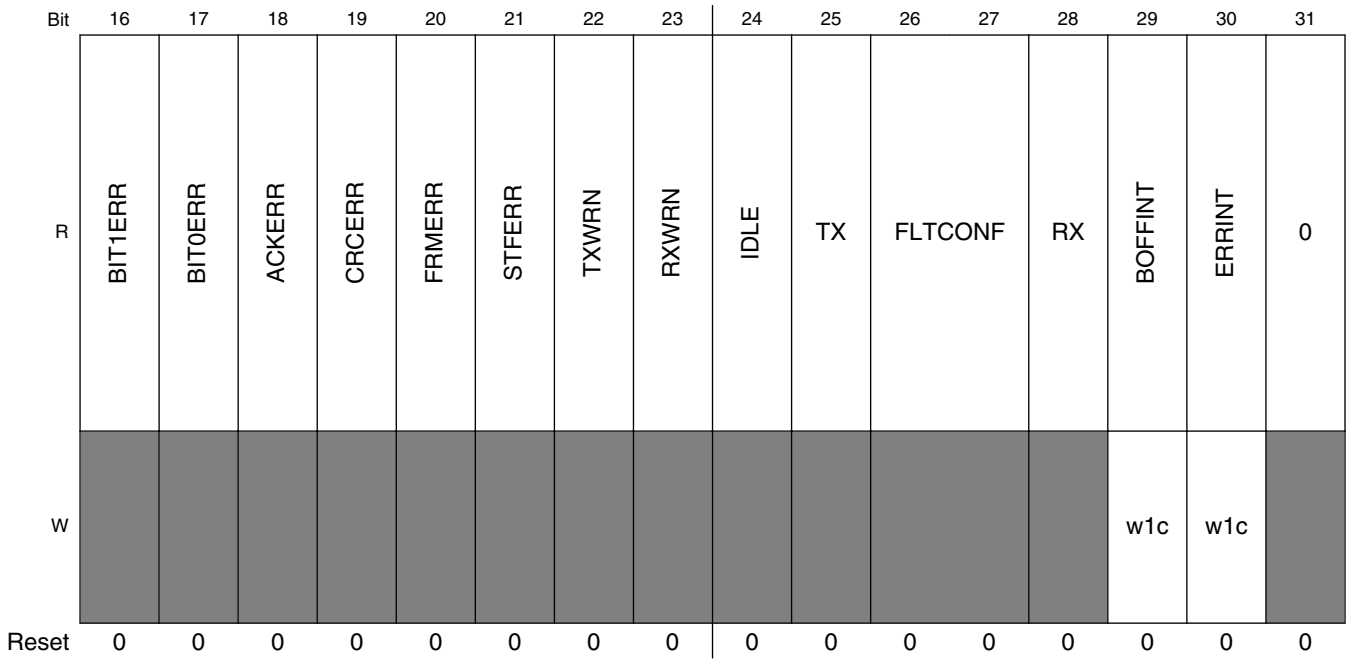
- Read this register to capture all error condition and status bits. This action clear the respective bits that were set since the last read access.
- Write 1 to clear the interrupt bit that has triggered the interrupt request.
- Write 1 to clear the ERR_OVR bit if it is set.

Starting from all error flags cleared, a first error event sets either the ERRINT or the ERRINT_FAST (provided the corresponding mask bit is asserted). If other error events in subsequent frames happen before the CPU to serve the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames had accumulated.

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BIT1ERR_FAST	BIT0ERR_FAST	0	CRCERR_FAST	FRMERR_FAST	STFERR_FAST	0				ERROVR	ERRINT_FAST	Reserved	SYNCH	TWRNINT	RWRNINT
W											w1c	w1c			w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
0 BIT1ERR_FAST	Bit1 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
1 BIT0ERR_FAST	Bit0 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 CRCERR_FAST	Cyclic Redundancy Check Error in the CRC field of CAN FD frames with the BRS bit set This bit indicates that a CRC Error has been detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
4 FRMERR_FAST	Form Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a Form Error has been detected by the receiver node in the Data Phase of CAN FD frames with the BRS bit set, that is, a fixed-form bit field contains at least one illegal bit.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence. 1 A Form Error occurred since last read of this register.
5 STFERR_FAST	Stuffing Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a Stuffing Error has been detected in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
6–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ERROVR	Error Overrun bit This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1. 0 Overrun has not occurred. 1 Overrun has occurred.
11 ERRINT_FAST	Error Interrupt for errors detected in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that at least one of the Error Bits detected in the Data Phase of CAN FD frames with the BRS bit set (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST or STFERR_FAST) is set. If the corresponding mask bit CAN_CTRL2[ERRMSK_FAST] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit detected in the Data Phase of CAN FD frames with the BRS bit set.
12 Reserved	This field is reserved.
13 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
14 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control 1 Register (CAN_CTRL1[TWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
15 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control 1

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	Register (CAN_CTRL1[RWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
16 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
17 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
18 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence. 1 An ACK error occurred since last read of this register.
19 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node either in a non-FD message or in the arbitration or data phase of a frame in CAN FD format, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
20 FRMERR	Form Error This bit indicates that a Form Error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence. 1 A Form Error occurred since last read of this register.
21 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
22 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission and is affected by the value of TXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. 0 No such occurrence. 1 TXERRCNT is greater than or equal to 96.
23 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception and is affected by the value of RXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. 0 No such occurrence. 1 RXERRCNT is greater than or equal to 96.
24 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence. 1 CAN bus is now IDLE.
25 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
26–27 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module. If the LOM bit in the Control Register 1 is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to CAN_ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, FLTCONF reports "Error Passive". 00 Error Active 01 Error Passive 1x Bus Off
28 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
29 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register 1 (CAN_CTRL1[BOFFMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence. 1 FlexCAN module entered Bus Off state.
30 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (BIT1ERR, BIT0ERR, ACKERR. CRCERR, FRMERR or STFERR) is set. If the corresponding mask bit CAN_CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit in the Error and Status Register.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.10 Interrupt Masks 2 register (CAN_IMASK2)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG2 bit is set.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF63TO32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK2 field descriptions

Field	Description
0–31 BUF63TO32M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. **NOTE:** Setting or clearing a bit in the CAN_IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

53.4.11 Interrupt Masks 1 register (CAN_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG1 bit is set.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF31TO0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK1 field descriptions

Field	Description
0–31 BUF31TO0M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the CAN_IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

53.4.12 Interrupt Flags 2 register (CAN_IFLAG2)

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG2 bit. If the corresponding CAN_IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG2 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF63TO32I																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG2 field descriptions

Field	Description
0–31 BUF63TO32I	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB63 to MB32. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

53.4.13 Interrupt Flags 1 register (CAN_IFLAG1)

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG1 bit. If the corresponding CAN_IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit CAN_MCR[RFEN] is set, the function of the 8 least significant interrupt flags changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO and the BUF4I to BUF0I field is reserved.

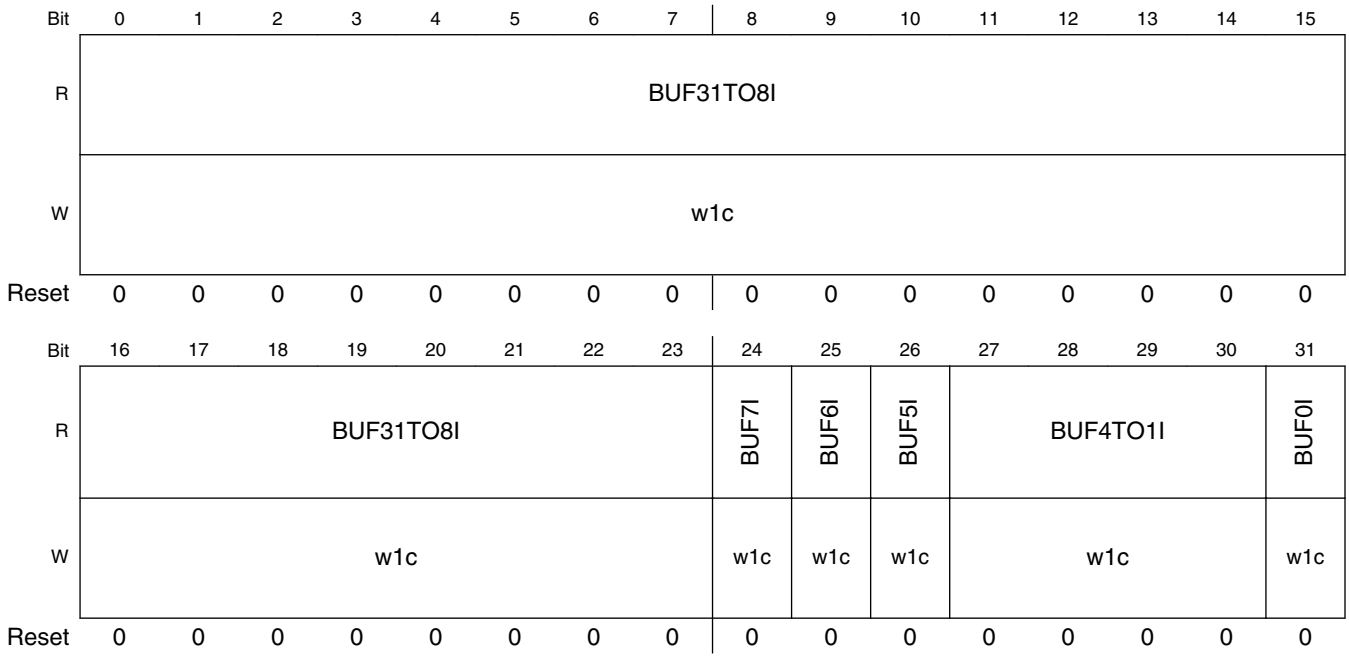
Before enabling the CAN_MCR[RFEN], the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the CAN_MCR[RFEN] bit is negated, the FIFO flags must be cleared. The same care must be taken when an CAN_CTRL2[RFFN] value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

FIFO must be disabled when FDEN bit in CAN_MCR register is enabled.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG1 bits whose MB value is greater than the CAN_MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Memory map/register definition

Address: 0h base + 30h offset = 30h



CAN_IFLAG1 field descriptions

Field	Description
0–23 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
24 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when CAN_MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
25 BUF6I	Buffer MB6 Interrupt Or "Rx FIFO Warning" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes.

Table continues on the next page...

CAN_IFLAG1 field descriptions (continued)

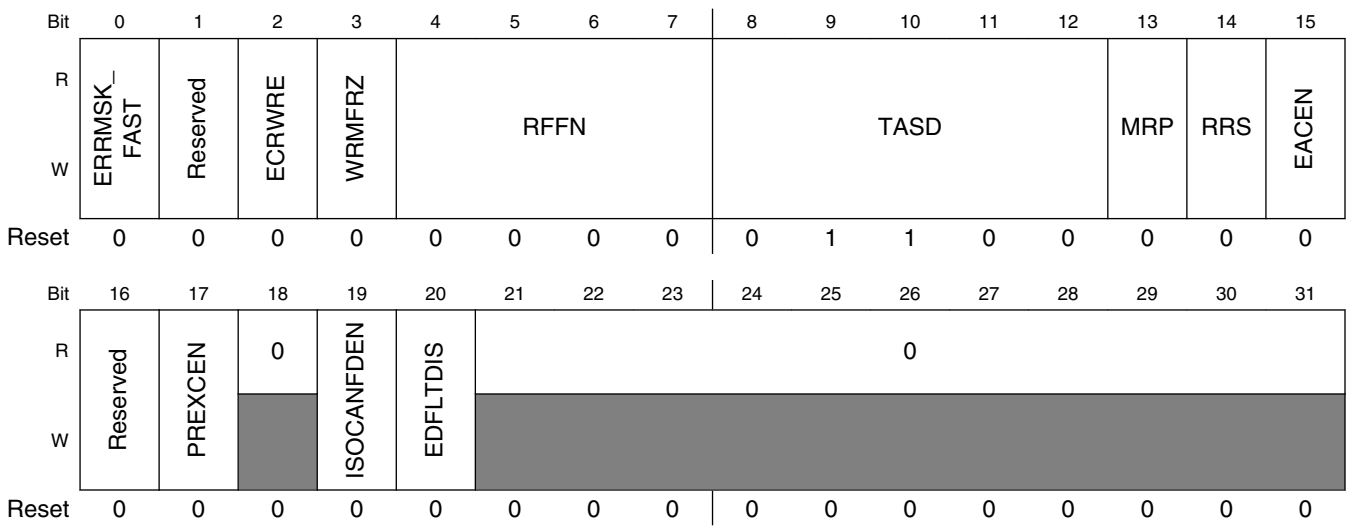
Field	Description
	The BUF6I flag represents "Rx FIFO Warning" when CAN_MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
26 BUF5I	Buffer MB5 Interrupt Or "Frames available in Rx FIFO" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF5I flag represents "Frames available in Rx FIFO" when CAN_MCR[RFEN] is set. In this case, the flag indicates that at least one frame is available to be read from the Rx FIFO. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1
27–30 BUF4TO1I	Buffer MB $_i$ Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when CAN_MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
31 BUF0I	Buffer MB0 Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF0I flag is reserved when CAN_MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

53.4.14 Control 2 register (CAN_CTRL2)

This register complements Control1 Register providing control bits for memory write access in Freeze Mode, for extending FIFO filter quantity, and for adjust the operation of internal FlexCAN processes like matching and arbitration.

The contents of this register are not affected by soft reset.

Address: 0h base + 34h offset = 34h



CAN_CTRL2 field descriptions

Field	Description
0 ERRMSK_FAST	Error Interrupt Mask for errors detected in the Data Phase of fast CAN FD frames This bit provides a mask for the ERRINT_FAST Interrupt in CAN_ESR1 register. 0 ERRINT_FAST Error interrupt disabled. 1 ERRINT_FAST Error interrupt enabled.
1 Reserved	This field is reserved. When writing to this field, always write the reset value.
2 ECRWRE	Error-correction Configuration Register Write Enable This bit enables the CAN_MECR register to be updated. This bit is automatically set to 0 if the protocol described in section Detection and Correction of Memory Errors is not followed. 0 Disable update. 1 Enable update.
3 WRMFRZ	Write-Access To Memory In Freeze Mode Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. CAN_MCR[RFEN] bit must not be set during FlexCAN memory initialization.

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	0 Maintain the write access restrictions. 1 Enable unrestricted write access to FlexCAN memory.
4–7 RFFN	Number Of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by CAN_MCR[MAXMB]. **NOTE:** Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows: (SETUP_MB - 6) × 4 where SETUP_MB is the least between the parameter NUMBER_OF_MB and CAN_MCR[MAXMB]. The number of remaining Mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2) If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional. **NOTE:** • The number of the last remaining available mailboxes is defined by the least value between the NUMBER_OF_MB minus 1 and the CAN_MCR[MAXMB] field. • If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

RFFN[3:0]	Number of Rx FIFO filter elements	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask
0x0	8	MB 0-7	MB 8-95	Elements 0-7	none
0x1	16	MB 0-9	MB 10-95	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-95	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-95	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-95	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-95	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-95	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-95	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-95	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-95	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-95	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-95	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-95	Elements 0-31	Elements 32-103

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filter elements</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask</th></tr>
<tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-95</td><td>Elements 0-31</td><td>Elements 32-111</td></tr>
<tr><td>0xE</td><td>120</td><td>MB 0-35</td><td>MB 36-95</td><td>Elements 0-31</td><td>Elements 32-119</td></tr>
<tr><td>0xF</td><td>128</td><td>MB 0-37</td><td>MB 38-95</td><td>Elements 0-31</td><td>Elements 32-127</td></tr>
</table>
</td></tr>
<tr><td>8–12
TASD</td><td>Tx Arbitration Start Delay

This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See Tx Arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes.</td></tr>
<tr><td>13
MRP</td><td>Mailboxes Reception Priority

If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Matching starts from Rx FIFO and continues on Mailboxes.
1 Matching starts from Mailboxes and continues on Rx FIFO.</td></tr>
<tr><td>14
RRS</td><td>Remote Request Storing

If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated.

If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID.

This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Remote Response Frame is generated.
1 Remote Request Frame is stored.</td></tr>
<tr><td>15
EACEN</td><td>Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes

This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Rx Mailbox filter’s IDE bit is always compared and RTR is never compared despite mask bits.
1 Enables the comparison of both Rx Mailbox filter’s IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.</td></tr>
<tr><td>16
Reserved</td><td>This field is reserved.
When writing to this field, always write the reset value.</td></tr>
<tr><td>17
PREXCEN</td><td>Protocol Exception Enable

This bit enables the Protocol Exception feature.

This field is writable only in Freeze mode.</td></tr>
</table>

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	0 Protocol Exception is disabled. 1 Protocol Exception is enabled.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 ISOCANFDEN	ISO CAN FD Enable This field enables the CAN FD protocol according to ISO specification (ISO 11898-1) (see CAN FD ISO compliance). This field is writable only in Freeze mode. 0 FlexCAN operates using the non-ISO CAN FD protocol. 1 FlexCAN operates using the ISO CAN FD protocol (ISO 11898-1).
20 EDFLTDIS	Edge Filter Disable This bit disables the Edge Filter used during the bus integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The Edge Filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD Frame) from being mistaken for an idle condition. This field is writable only in Freeze mode. 0 Edge Filter is enabled. 1 Edge Filter is disabled.
21–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.15 Error and Status 2 register (CAN_ESR2)

This register reports some general status information.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

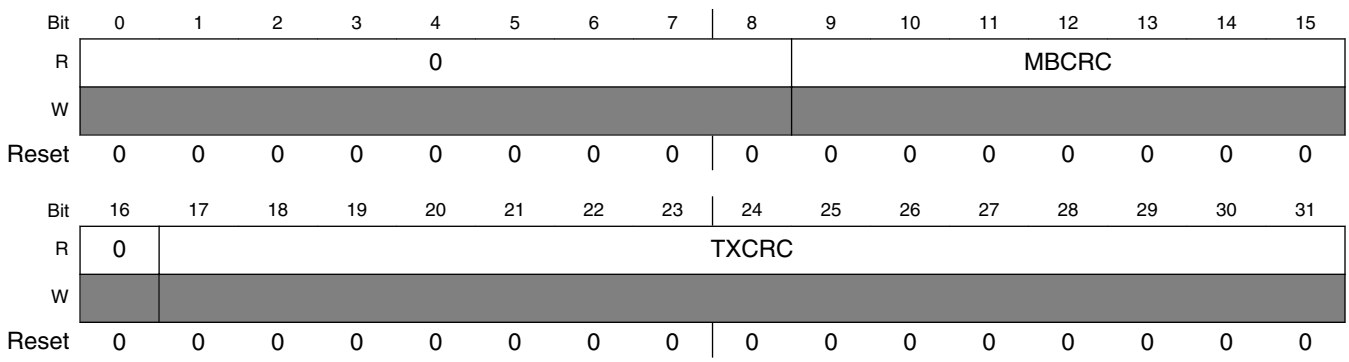
CAN_ESR2 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LPTM	Lowest Priority Tx Mailbox If CAN_ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the CAN_ESR2[IMB] bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CAN_CTRL1[LBUF] bit value. If CAN_CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CAN_CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VPS	Valid Priority Status This bit indicates whether CAN_ESR2[IMB] and CAN_ESR2[LPTM] contents are currently valid or not. It is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. **NOTE:** CAN_ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When CAN_MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with CAN_IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
18 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: • During arbitration, if an CAN_ESR2[LPTM] is found and it is inactive. • If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** CAN_ESR2[LPTM] mechanism have the following behavior: if an MB is successfully transmitted and CAN_ESR2[IMB]=0 (no inactive Mailbox), then CAN_ESR2[VPS] and CAN_ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into CAN_ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages for non FD messages. This register only reports the 15 low order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, the CAN_FDCRC register must be used. This register is updated at the same time the Tx Interrupt Flag is asserted.

Address: 0h base + 44h offset = 44h



CAN_CRCR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in CAN_CRCR[TXCRC] field.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 TXCRC	Transmitted CRC value This field indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in CAN_FDCRC register.

53.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled, RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CAN_CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FGM																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFGMASK field descriptions

Field	Description
0–31 FGM	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment. The following table shows how the FGM bits correspond to each IDAF field.

Rx FIFO ID Filter Table Elements Format (CAN_MCR[IDAM])	Identifier Acceptance Filter Fields					
	RTR	IDE	RXIDA	RXIDB [1]	RXIDC [2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]		-
C	-	-		-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	

1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

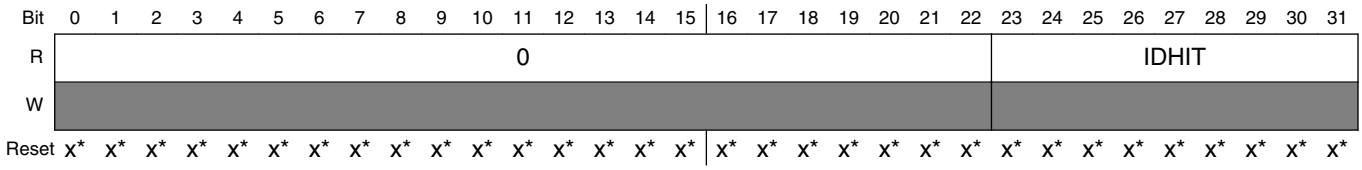
1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

53.4.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 0h base + 4Ch offset = 4Ch



* Notes:
- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the CAN_IFLAG1[BUF5I] is asserted.

53.4.19 CAN Bit Timing Register (CAN_CBT)

This register is an alternative way to store the CAN bit timing variables described in CAN_CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW are extended versions of PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW bit fields respectively.

The BTF bit selects the use of the timing variables defined in this register.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

NOTE

When the CAN FD feature is enabled (CAN_MCR[FDEN] is set), always set CAN_CBT[BTF].

Memory map/register definition

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BTF	EPRESDIV										ERJW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EPROPSEG						EPSEG1					EPSEG2				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CBT field descriptions

Field	Description
0 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW replacing the CAN bit timing variables defined in CAN_CTRL1 register. This field can be written in Freeze mode only. 0 Extended bit time definitions disabled. 1 Extended bit time definitions enabled.
1–10 EPRESDIV	Extended Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PRESDIV] value range. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (EPRESDIV + 1)
11–15 ERJW	Extended Resync Jump Width This 5-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[RJW] value range. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = ERJW + 1.
16–21 EPROPSEG	Extended Propagation Segment This 6-bit field defines the length of the Propagation Segment in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.
22–26 EPSEG1	Extended Phase Segment 1 This 5-bit field defines the length of Phase Segment 1 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

Table continues on the next page...

CAN_CBT field descriptions (continued)

Field	Description
27–31 EPSEG2	Extended Phase Segment 2 This 5-bit field defines the length of Phase Segment 2 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

53.4.20 Interrupt Masks 3 Register (CAN_IMASK3)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB95 to MB64. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG3 bit is set.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF95TO64M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK3 field descriptions

Field	Description
0–31 BUF95TO64M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB95 to MB64. When CAN FD is enabled, the MB range is defined in accordance to the MBDSRs bit fields of CAN_FDCTRL register. **NOTE:** Setting or clearing a bit in the CAN_IMASK3 Register can assert or negate an interrupt request, if the corresponding CAN_IFLAG3 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

53.4.21 Interrupt Flags 3 Register (CAN_IFLAG3)

This register defines the flags for the 32 Message Buffer interrupts for MB95 to MB64. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG3 bit. If the corresponding CAN_IMASK3 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG3 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF95TO64																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG3 field descriptions

Field	Description
0–31 BUF95TO64	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB95 to MB64.When CAN FD is enabled, the MB range is defined in accordance to the MBDSRs bit fields of CAN_FDCTRL register. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception 1 The corresponding buffer has successfully completed transmission or reception

53.4.22 Rx Individual Mask Registers (CAN_RXIMR*n*)

The RX Individual Mask Registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CAN_MCR[RFEN] bit is negated), an individual mask is provided for each available Rx Mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CAN_MCR[RFEN] bit is asserted), an individual mask is provided for each Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of CAN_CTRL2[RFFN] (see Rx FIFO).

CAN_RXIMR0 stores the individual mask associated to either MB0 or ID Filter Table Element 0, CAN_RXIMR1 stores the individual mask associated to either MB1 or ID Filter Table Element 1 and so on.

CAN_RXIMR registers can only be accessed by the CPU while the module is in Freeze mode, otherwise, they are blocked by hardware. These registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

It is possible for the RXIMR memory region to be accessed as general purpose memory. See Bus interface for more information.

Address: 0h base + 880h offset + (4d × i), where i=0d to 95d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXIMR*n* field descriptions

Field	Description
0–31 MI	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

53.4.23 Memory Error Control Register (CAN_MECR)

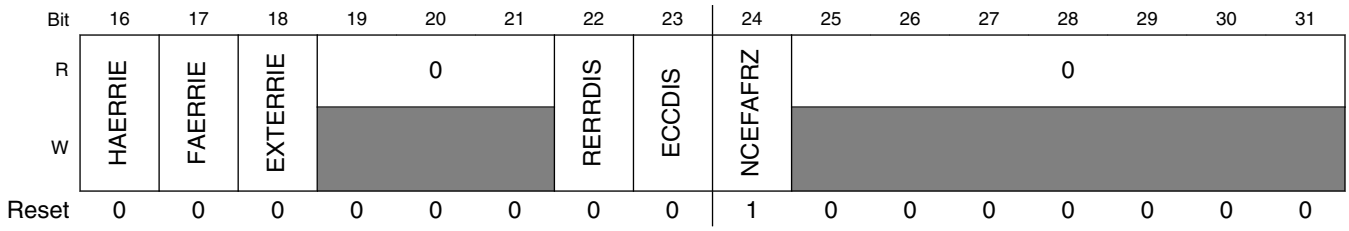
This register contains control bits for memory error detection and correction (ECC).

NOTE

When bit CTRL2[ECRWRE] is 0, writes in this register are blocked, except in the ECRWRDIS bit.

Address: 0h base + AE0h offset = AE0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ECRWRDIS	0											Reserved	Reserved	0	Reserved
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0



CAN_MECR field descriptions

Field	Description
0 ECRWRDIS	Error Configuration Register Write Disable Disables writes on this register. This bit is automatically set to 1 (disabled) when CTRL2[ECRWRE] is enabled. The protocol described in section Detection and Correction of Memory Errors must be followed. 0 Write is enabled. 1 Write is disabled.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. When writing to this field, always write the reset value.
13 Reserved	This field is reserved. When writing to this field, always write the reset value.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. When writing to this field, always write the reset value.
16 HAERRIE	Host Access Error Injection Enable Enables the injection of errors only in memory reads issued by the host (CPU). 0 Injection is disabled. 1 Injection is enabled.
17 FAERRIE	FlexCAN Access Error Injection Enable Enables the injection of errors only in memory reads issued by the FlexCAN internal processes. 0 Injection is disabled. 1 Injection is enabled.
18 EXTERRIE	Extended Error Injection Enable Memory accesses performed by internal FlexCAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See Error Injection Data Pattern Register (CAN_FCERRIDPR) and Error Injection Parity Pattern Register (CAN_FCERRIPPR) 0 Error injection is applied only to the 32-bit word. 1 Error injection is applied to the 64-bit word.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_MECR field descriptions (continued)

Field	Description
22 RERRDIS	Error Report Disable Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active. **NOTE:** When reading the report registers, this bit must be set to assure coherence on the consecutive register reads. 0 Enable updates of the error report registers. 1 Disable updates of the error report registers.
23 ECCDIS	Error Correction Disable Disables completely the memory detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions. 0 Enable memory error correction. 1 Disable memory error correction.
24 NCEFAFRZ	Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode Determines the response when a non-correctable error is detected in a memory read performed by FlexCAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by FlexCAN internal processes. 0 Keep normal operation. 1 Put FlexCAN in Freeze mode (according to the "Freeze mode" section).
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.24 Error Injection Address Register (CAN_ERRIAR)

This register holds the address where error is to be injected.

Use the following table to convert from the memory map address to the location in the physical FlexCAN RAM (where pairs of values are provided, the first is the address for CAN_MCR[FDEN] negated, the second is for CAN_MCR[FDEN] asserted):

RAM contents	Injection address	Memory map
FlexCAN Registers	Not mapped	-
MBs	0x0000	0x0080
Reserved	-	0x0480
RXIMRs	0x0600	0x0880
Reserved	-	0x0980
RXFIR_0	0x0780	0x0A80
RXFIR_1	0x0784	0x0A84
RXFIR_2	0x0788	0x0A88

Table continues on the next page...

RAM contents	Injection address	Memory map
RXFIR_3	0x078C	0x0A8C
RXFIR_4	0x0790	0x0A90
RXFIR_5	0x0794	0x0A94
Reserved	-	0x0A98
RXMGMASK	0x07A0	0x0AA0
RXFGMASK	0x07A4	0x0AA4
RX14MASK	0x07A8	0x0AA8
RX15MASK	0x07AC	0x0AAC
Tx_SMB	0x07B0	0x0AB0 / 0x0F28
Rx_SMB0	0x07C0 / 0x07F8	0x0AC0 / 0x0F70
Rx_SMB1	0x07D0 / 0x0840	0x0AD0 / 0x0FB8
ECC Registers	Not mapped	0x0AE0
Reserved	-	0x0B00
CAN FD Registers	Not mapped	0x0C00
Reserved	-	0x0C0C

Address: 0h base + AE4h offset = AE4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJADDR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIAR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 INJADDR	Error Injection Address This bit field defines the physical RAM address where error is to be injected (see table above). **NOTE:** The two least significant bits are always read as zero.

53.4.25 Error Injection Data Pattern Register (CAN_ERRIDPR)

Holds the error pattern to be injected in the data word read from memory.

Address: 0h base + AE8h offset = AE8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIDPR field descriptions

Field	Description
0–31 DFLIP	Data flip pattern Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.

53.4.26 Error Injection Parity Pattern Register (CAN_ERRIPPR)

Holds the error pattern to be injected in parity bits read from memory along with data word.

Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.

Address: 0h base + AECh offset = AECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			PFLIP3					0			PFLIP2					0			PFLIP1					0			PFLIP0				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIPPR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP3	Parity Flip Pattern For Byte 3 (most significant)
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP2	Parity Flip Pattern For Byte 2
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP1	Parity Flip Pattern For Byte 1
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP0	Parity Flip Pattern For Byte 0 (Least Significant)

53.4.27 Error Report Address Register (CAN_RERRAR)

Reports the address used for an access in which an error (correctable or non-correctable) was detected, and also reports the identification of the source of that access.

This address is always reported using a 32-bit alignment. Non-aligned accesses (ERRADDR[1:0] non-0) are reported with the address aligned and data is reported in RERRDR accordingly shifted. In case of errors detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word which the error was detected is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

Address: 0h base + AF0h offset = AF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							NCE	0					SAID		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERRADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NCE	Non-Correctable Error Indicates that the report is due to an non-correctable error. 0 Reporting a correctable-error 1 Reporting a non-correctable error
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SAID	SAID[2] — Identification of the requestor of the memory read request: • 0 = Requested by FlexCAN internal processes • 1 = Requested by Host (CPU) SAID[1] — Details of FlexCAN operation: • 0 = Move • 1 = Scanning SAID[0] — Operation that requested the memory read: • 0 = Transmission • 1 = Reception

Table 53-5. Source of memory access

SAID[2:0]	Error during...
0	Move-out FlexCAN access
1	Move-in

Table continues on the next page...

CAN_RERRAR field descriptions (continued)

Field	Description
	Table 53-5. Source of memory access (continued)

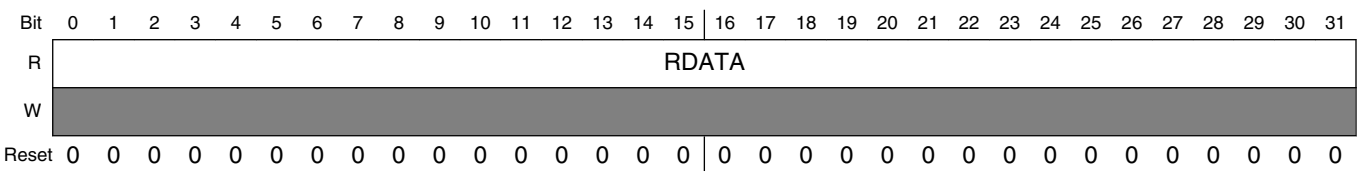
SAID[2:0]	Error during...
2	Tx Arbitration
3	Rx Matching
4	Move-out Host access
5-7	Reserved

16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 ERRADDR	Address Where The Error Was Detected See the description of the Error Injection Address Register (ERRIAR).

53.4.28 Error Report Data Register (CAN_RERRDR)

Reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit (see the “Message Buffer Code for Rx buffers” table) when reading a Message Buffer.

Address: 0h base + AF4h offset = AF4h



CAN_RERRDR field descriptions

Field	Description
0–31 RDATA	Raw data word read from memory with error

53.4.29 Error Report Syndrome Register (CAN_RERRSYNR)

Holds the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. (SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.)

Table 53-6. Syndrome Definition

SYNDn (hex)	Type	Bit affected
00	-	none (no error)
01	Code	0
02	Code	1
04	Code	2
07	Data	5
08	Code	3
0E	Data	7
10	Code	4
13	Data	2
15	Data	6
16	Data	1
19	Data	3
1A	Data	4
1C	Data	0
06	-	All-zeros non-correctable error
1F	-	All-ones non-correctable error
All others	-	Non-correctable error

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated (see the "Error Indication" section) and reported (see the "Error Reporting" section).

Address: 0h base + AF8h offset = AF8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0		SYND3					BE2	0		SYND2				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0		SYND1					BE0	0		SYND0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

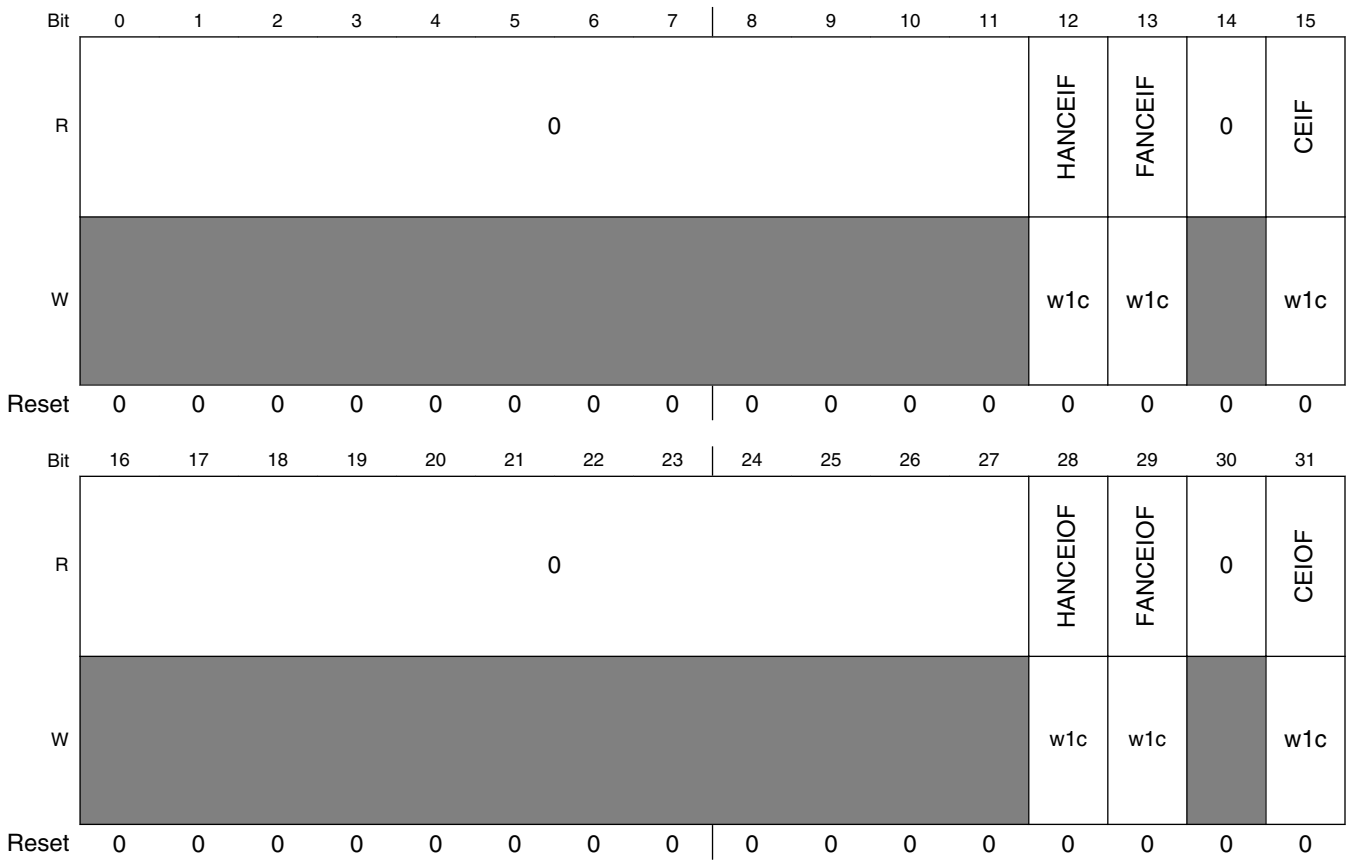
CAN_RERRSYNR field descriptions

Field	Description
0 BE3	Byte Enabled For Byte 3 (Most Significant) 0 The byte was not read. 1 The byte was read.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome For Byte 3 (Most Significant) See the "Syndrome Definition" table.
8 BE2	Byte Enabled For Byte 2 0 The byte was not read. 1 The byte was read.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome For Byte 2 See the "Syndrome Definition" table.
16 BE1	Byte Enabled For Byte 1 0 The byte was not read. 1 The byte was read.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome for Byte 1 See the "Syndrome Definition" table.
24 BE0	Byte Enabled For Byte 0 (least significant) 0 The byte was not read. 1 The byte was read.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome For Byte 0 (least significant) See the "Syndrome Definition" table.

53.4.30 Error Status Register (CAN_ERRSR)

Holds the status bits of the error correction and detection operations. These flags can be cleared by writing 1 to them. Writing 0 has no effect.

Address: 0h base + AFCh offset = AFCh



CAN_ERRSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 HANCEIF	Host Access With Non-Correctable Error Interrupt Flag Indicates that a non-correctable error was detected in a memory read initiated by Host. A bus transfer error is asserted for that access. No interrupt is associated to this flag. 0 No non-correctable errors were detected in Host accesses so far. 1 A non-correctable error was detected in a Host access.
13 FANCEIF	FlexCAN Access With Non-Correctable Error Interrupt Flag

Table continues on the next page...

CAN_ERRSR field descriptions (continued)

Field	Description
	Indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes. No interrupt is associated to this flag. 0 No non-correctable errors were detected in FlexCAN accesses so far. 1 A non-correctable error was detected in a FlexCAN access.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEIF	Correctable Error Interrupt Flag Indicates that a correctable error was detected in a memory read. 0 No correctable errors were detected so far. 1 A correctable error was detected.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 HANCEIOF	Host Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by Host when HANCEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on non-correctable errors in Host access 1 Overrun on non-correctable errors in Host access
29 FANCEIOF	FlexCAN Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes when FANCEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on non-correctable errors in FlexCAN access 1 Overrun on non-correctable errors in FlexCAN access
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEIOF	Correctable Error Interrupt Overrun Flag Indicates that a correctable error was detected in a memory read when CEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on correctable errors 1 Overrun on correctable errors

53.4.31 CAN FD Control Register (CAN_FDCTRL)

This register contains control bits for the CAN FD operation. It also defines the data size of Message Buffers allocated in different partitions of RAM (memory blocks) as described in the table below.

When 8 bytes payload is selected:

- Block R0 allocates MB0 to MB31.

- Block R1 allocates MB32 to MB63.
- Block R2 allocates MB64 to MB95.

When more than 8 bytes payload is selected, the maximum number of MBs in a block is limited as described below:

Table 53-7. Number of Message Buffers

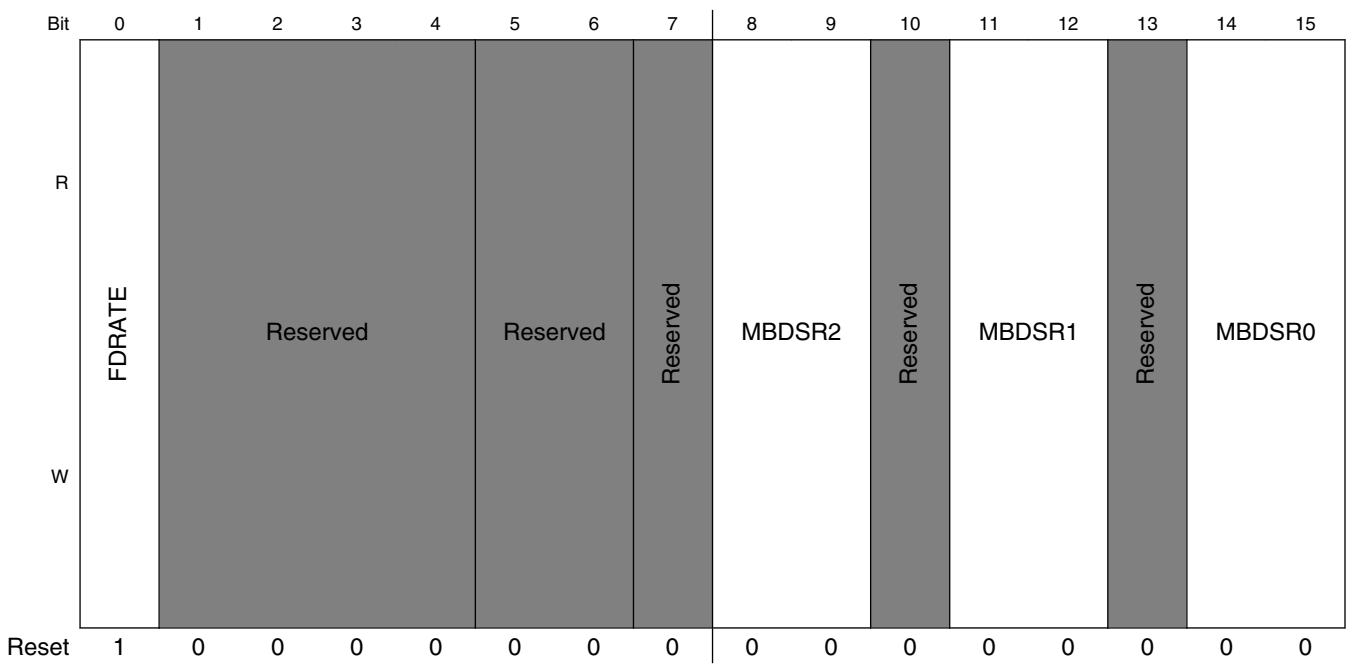
Payload Size	Maximum number of Message Buffers per RAM block
8 bytes	32
16 bytes	21
32 bytes	12
64 bytes	7

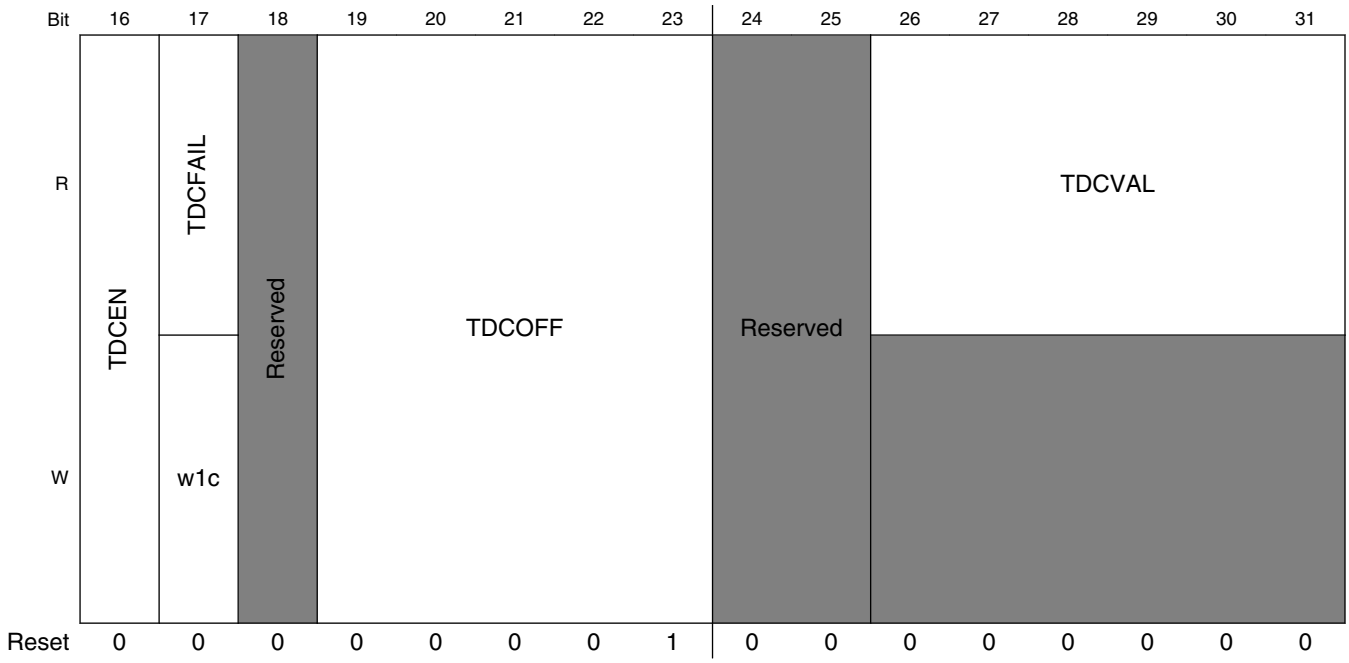
NOTE

One memory block fits exactly 32 MBs with 8 bytes payload. For the other options of payload sizes, empty memory may exist between last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of this register are not affected by soft reset.

Address: 0h base + C00h offset = C00h





CAN_FDCTRL field descriptions

Field	Description
0 FDRATE	Bit Rate Switch Enable This bit enables the effect of the Bit Rate Switch (BRS bit) during the data phase of Tx messages. The CPU can write this bit any time. However, its effect turns active only when the CAN bus is in Wait for Bus Idle, Bus Idle or Bus Off state, or when the current frame under reception or transmission reaches the interframe space. By negating the CAN_FDCTRL[FDRATE] bit, the CPU can force all bits in CAN FD messages to be transmitted in nominal bit rate, despite of the value in the BRS bit of the Tx MBs. 0 Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect. 1 Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive.
1–4 Reserved	This field is reserved.
5–6 Reserved	This field is reserved.
7 Reserved	This field is reserved.
8–9 MBDSR2	Message Buffer Data Size for Region 2 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R2 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.

Table continues on the next page...

CAN_FDCTRL field descriptions (continued)

Field	Description
10 Reserved	This field is reserved.
11–12 MBDSR1	Message Buffer Data Size for Region 1 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R1 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.
13 Reserved	This field is reserved.
14–15 MBDSR0	Message Buffer Data Size for Region 0 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R0 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.
16 TDCEN	Transceiver Delay Compensation Enable This bit can be used to enable and disable the TDC feature. It can be written in Freeze mode only. **NOTE:** TDC must be disabled when the Loop Back Mode is enabled (see CAN_CTRL1[LPB] register). 0 TDC is disabled 1 TDC is enabled
17 TDCFAIL	Transceiver Delay Compensation Fail This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones (see Transceiver Delay Compensation. TDCFAIL sets in the first time FlexCAN detects the out of range condition. The CPU needs to write 1 to clear it. 0 Measured loop delay is in range. 1 Measured loop delay is out of range.
18 Reserved	This field is reserved.
19–23 TDCOFF	Transceiver Delay Compensation Offset This bit field contains the offset value to be added to the measured transceiver's loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver Delay Compensation for more details on how the loop delay measurement is performed. TDCOFF can be written in Freeze mode only. Its value can be defined in Protocol Engine (PE) Clock periods (CANCLK, see Protocol timing for more details), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation.

Table continues on the next page...

CAN_FDCTRL field descriptions (continued)

Field	Description
	NOTE: It is not recommended to use TDCOFF equal to zero.
24–25 Reserved	This field is reserved.
26–31 TDCVAL	Transceiver Delay Compensation Value This register contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the TDCOFF value specified in the CAN_FDCTRL register. This value is an integer multiple of the Protocol Engine (PE) Clock period (CANCLK). See Protocol timing for more details on how the loop delay measurement is performed.

53.4.32 CAN FD Bit Timing Register (CAN_FDCBT)

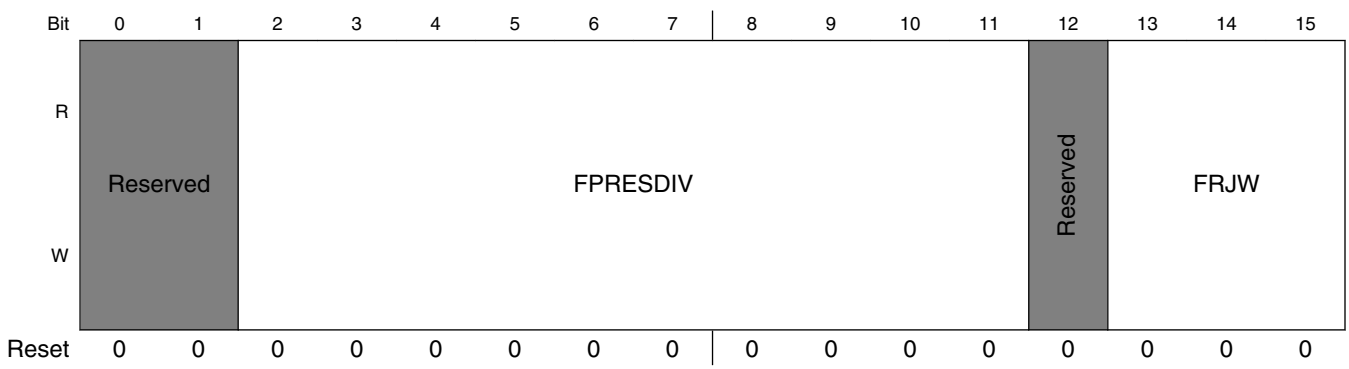
This register stores the CAN bit timing variables used in the data phase of CAN FD messages when the CAN_FDCTRL[FDRATE] is set, compatible with CAN FD specification. FPRESDIV, FPROPSEG, FPSEG1, FPSEG2 and FRJW are used to define the time quantum duration, the number of time quanta per CAN bit and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

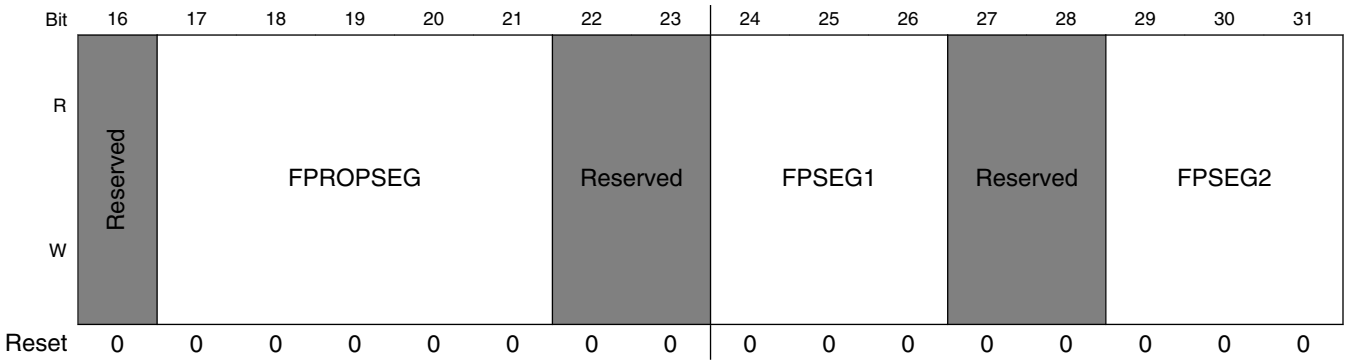
The contents of this register are not affected by soft reset.

NOTE

The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

Address: 0h base + C04h offset = C04h





CAN_FDCBT field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–11 FPRESDIV	Fast Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (FPRESDIV + 1). **NOTE:** To minimize errors when processing FD frames, use the same value for FPRESDIV and PRESDIV (in CAN_CBT or CAN_CTRL1). For more details refer to the first NOTE in section CAN FD frames.
12 Reserved	This field is reserved.
13–15 FRJW	Fast Resync Jump Width This 3-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization in the data bit rate portion of a CAN FD message with the BRS bit set. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = FSJW + 1.
16 Reserved	This field is reserved.
17–21 FPROPSEG	Fast Propagation Segment This 5-bit field defines the length of the Propagation Segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = FPROPSEG × Time-Quanta. Time-Quantum = one Sclock period.
22–23 Reserved	This field is reserved.
24–26 FPSEG1	Fast Phase Segment 1

Table continues on the next page...

CAN_FDCBT field descriptions (continued)

Field	Description
	This 3-bit field defines the length of Phase Segment 1 in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 1 = (FPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
27–28 Reserved	This field is reserved.
29–31 FPSEG2	Fast Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 2 = (FPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

53.4.33 CAN FD CRC Register (CAN_FDCRC)

This register provides information about the CRC of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats, as shown below.

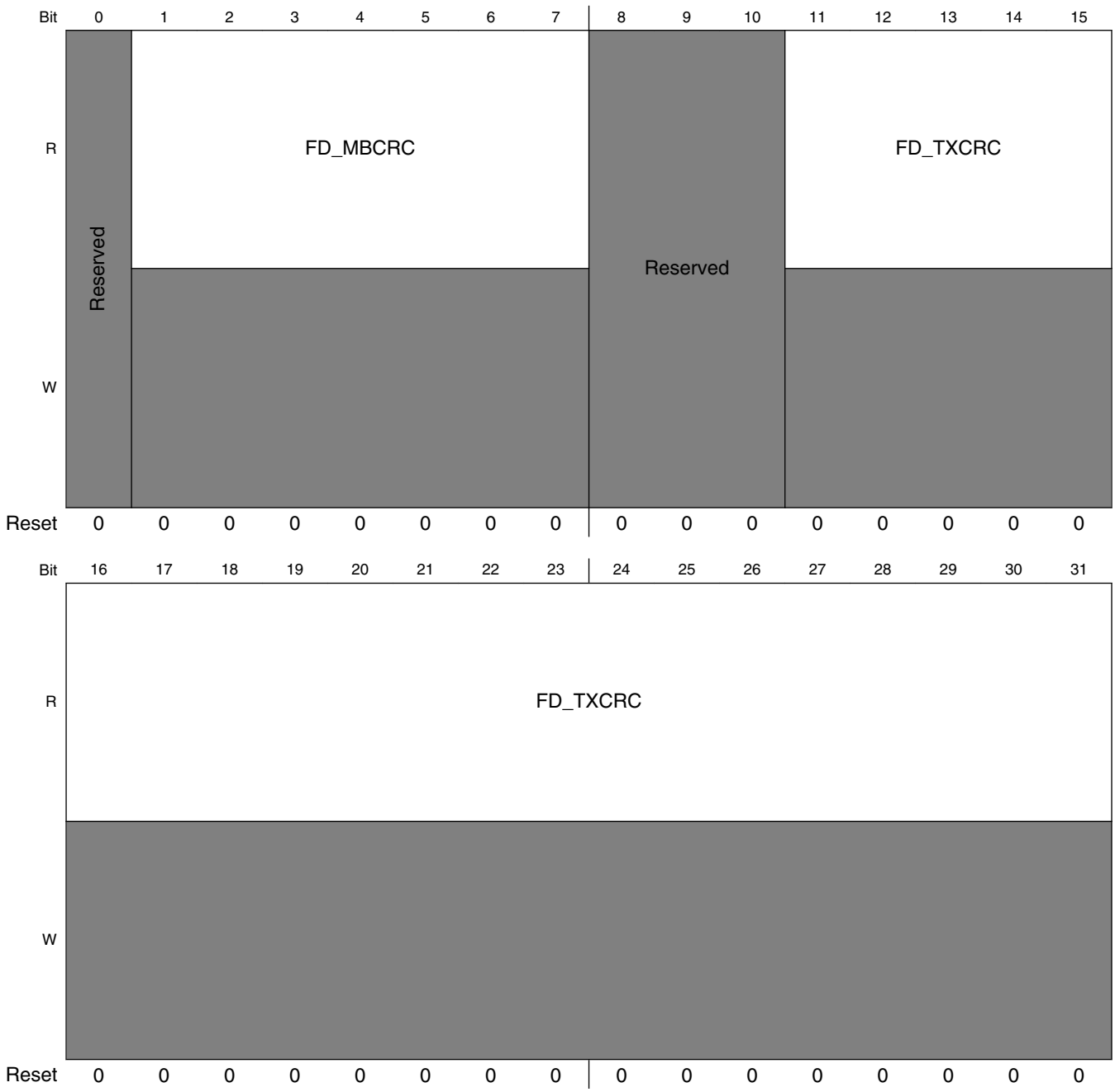
The CRC_15 polynomial is used for all frames in CAN format. The CRC_17 polynomial is used for frames in CAN FD format with a DATA FIELD up to sixteen bytes. The CRC_21 polynomial is used for frames in CAN FD format with a DATA FIELD longer than sixteen bytes. Each polynomial shown below results in a Hamming Distance of 6. This register is updated at the same time the Tx Interrupt Flag is asserted.

CRC_15 = 0xC599: $(x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1)$

CRC_17 = 0x3685B: $(x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x^3 + x^1 + 1)$

CRC_21 = 0x302899: $(x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1)$

Address: 0h base + C08h offset = C08h



CAN_FDCRC field descriptions

Field	Description
0 Reserved	This field is reserved.
1–7 FD_MBCRC	CRC Mailbox Number for FD_TXCRC This field indicates the number of the Mailbox corresponding to the value in FD_TXCRC field, for both FD and non-FD frames. It reports the same information as in MBCRC bit field in CAN_CRCR register.

Table continues on the next page...

CAN_FDCRC field descriptions (continued)

Field	Description
8–10 Reserved	This field is reserved.
11–31 FD_TXCRC	Extended Transmitted CRC value This 21-bit field contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the 6 most significant bits and the 4 most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as CRC Register.

53.4.34 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16, 24, 40 or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32 or 64 data bytes, respectively.

The memory area from 0x80 to 0x67F is used by the mailboxes. When CAN FD is enabled, the exact address for each MB depends on the size of its payload. See FlexCAN Memory Partition for CAN FD for more detailed information.

Table 53-8. Message buffer structure - example with 64 bytes payload

	0	1	2	3	4 7	8	9	10	11	12 13 14 15	16 23	24 31
0x0	EDL	BRS	ESI		CODE		SRR	IDE	RTR	DLC	TIME STAMP	
0x4	PRIO			ID (Standard/Extended)							ID (Extended)	
0x8	Data Byte 0					Data Byte 1					Data Byte 2	Data Byte 3
0xC	Data Byte 4					Data Byte 5					Data Byte 6	Data Byte 7
0x10	Data Byte 8					Data Byte 9					Data Byte 10	Data Byte 11
0x14	Data Byte 12					Data Byte 13					Data Byte 14	Data Byte 15
0x18	Data Byte 16					Data Byte 17					Data Byte 18	Data Byte 19
0x1C	Data Byte 20					Data Byte 21					Data Byte 22	Data Byte 23
0x20	Data Byte 24					Data Byte 25					Data Byte 26	Data Byte 27
0x24	Data Byte 28					Data Byte 29					Data Byte 30	Data Byte 31
0x28	Data Byte 32					Data Byte 33					Data Byte 34	Data Byte 35
0x2C	Data Byte 36					Data Byte 37					Data Byte 38	Data Byte 39

Table continues on the next page...

Table 53-8. Message buffer structure - example with 64 bytes payload (continued)

0x30	Data Byte 40	Data Byte 41	Data Byte 42	Data Byte 43
0x34	Data Byte 44	Data Byte 45	Data Byte 46	Data Byte 47
0x38	Data Byte 48	Data Byte 49	Data Byte 50	Data Byte 51
0x3C	Data Byte 52	Data Byte 53	Data Byte 54	Data Byte 55
0x40	Data Byte 56	Data Byte 57	Data Byte 58	Data Byte 59
0x44	Data Byte 60	Data Byte 61	Data Byte 62	Data Byte 63

= Unimplemented or Reserved

EDL - Extended Data Length

This bit distinguishes between CAN format and CAN FD format frames. The EDL bit must not be set for Message Buffers configured to RANSWER with code field 0b1010 (see Table below).

BRS - Bit Rate Switch

This bit defines whether the bit rate is switched inside a CAN FD format frame.

ESI - Error State Indicator

This bit indicates if the transmitting node is error active or error passive.

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 53-9 and Table 53-10. See Functional description for additional information.

Table 53-9. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE - MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL.

Table continues on the next page...

Table 53-9. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.

Table continues on the next page...

Table 53-9. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.[5]	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CAN_CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		-	-	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[6]	-	FULL	-	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		-	OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit, see "Control 2 Register (CAN_CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Code 0b1010 must be used in Message Buffers configured in CAN FD format, having the EDL bit set.
6. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 53-10. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in arbitration process.
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in arbitration process.
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE - ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 53-9, Table 53-10, and the description of the RRS bit in Control 2 Register (CAN_CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

NOTE

When configuring CAN FD frames, the RTR bit must be negated.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 53-8) . In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 53-11).

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is used only when LPRIO_EN bit is set in CAN_MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0 to 63 - Data Field

Up to sixty four bytes can be used for a data frame, depending on the size of payload selected for the Message Buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

Table 53-11. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8	DATA BYTE 0 to 7
9	DATA BYTE 0 to 11
10	DATA BYTE 0 to 15
11	DATA BYTE 0 to 19
12	DATA BYTE 0 to 23
13	DATA BYTE 0 to 31
14	DATA BYTE 0 to 47
15	DATA BYTE 0 to 63

53.4.35 FlexCAN Memory Partition for CAN FD

When CAN FD is enabled, the FlexCAN RAM can be partitioned in blocks of 512 bytes. Each block can accommodate a number of Message Buffers which depends on the configuration provided by CAN_FDCTRL[MBDSRn] bit fields as shown in table below.

Table 53-12. RAM partition

RAM block	Number of MBs with 8 bytes (default range)	Size control bit field in CAN_FDCTRL register	Number of MBs of different sizes, per block
0	0 to 31	MBDSR0	MBDSR0=00, 32 MBs with 8 bytes payload MBDSR0=01, 21 MBs with 16 bytes payload MBDSR0=10, 12 MBs with 32 bytes payload MBDSR0=11, 7 MBs with 64 bytes payload
1	32 to 63	MBDSR1	MBDSR1=00, 32 MBs with 8 bytes payload MBDSR1=01, 21 MBs with 16 bytes payload MBDSR1=10, 12 MBs with 32 bytes payload MBDSR1=11, 7 MBs with 64 bytes payload
2	64 to 95	MBDSR2	MBDSR2=00, 32 MBs with 8 bytes payload MBDSR2=01, 21 MBs with 16 bytes payload MBDSR2=10, 12 MBs with 32 bytes payload MBDSR2=11, 7 MBs with 64 bytes payload

When payload sizes of 16, 32 or 64 bytes are configured in some or all RAM blocks, the total number of MBs and its respective number order may differ from the default configuration of 8 bytes. For example, suppose Block0 is configured to 8 bytes payload, Block1 to 16 bytes, Block2 to 32 bytes, than the following table indicates how the Message Buffers will be arranged into RAM.

Table 53-13. RAM partition example

RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
0	CAN_FDCTRL[MBDSR0]=00, 8 bytes payload	32	0 to 31

Table continues on the next page...

Table 53-13. RAM partition example (continued)

RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
1	CAN_FDCTRL[MBDSR1]=01, 16 bytes payload	21	32 to 52
2	CAN_FDCTRL[MBDSR2]=10, 32 bytes payload	12	53 to 64

53.4.36 FlexCAN message buffer memory map

The FlexCAN memory buffers are allocated in memory according to the tables below.

Table 53-14. 8-byte message buffers

Address offset (hex)	MBDSR=b00 8-byte payload
0080	MB0
0090	MB1
00A0	MB2
00B0	MB3
00C0	MB4
00D0	MB5
00E0	MB6
00F0	MB7
0100	MB8
0110	MB9
0120	MB10
0130	MB11
0140	MB12
0150	MB13
0160	MB14
0170	MB15
0180	MB16
0190	MB17
01A0	MB18
01B0	MB19
01C0	MB20
01D0	MB21
01E0	MB22
01F0	MB23
0200	MB24

Table continues on the next page...

Table 53-14. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0210	MB25
0220	MB26
0230	MB27
0240	MB28
0250	MB29
0260	MB30
0270	MB31
0280	MB32
0290	MB33
02A0	MB34
02B0	MB35
02C0	MB36
02D0	MB37
02E0	MB38
02F0	MB39
0300	MB40
0310	MB41
0320	MB42
0330	MB43
0340	MB44
0350	MB45
0360	MB46
0370	MB47
0380	MB48
0390	MB49
03A0	MB50
03B0	MB51
03C0	MB52
03D0	MB53
03E0	MB54
03F0	MB55
0400	MB56
0410	MB57
0420	MB58
0430	MB59
0440	MB60
0450	MB61
0460	MB62

Table continues on the next page...

Table 53-14. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0470	MB63
0480	MB64
0490	MB65
04A0	MB66
04B0	MB67
04C0	MB68
04D0	MB69
04E0	MB70
04F0	MB71
0500	MB72
0510	MB73
0520	MB74
0530	MB75
0540	MB76
0550	MB77
0560	MB78
0570	MB79
0580	MB80
0590	MB81
05A0	MB82
05B0	MB83
05C0	MB84
05D0	MB85
05E0	MB86
05F0	MB87
0600	MB88
0610	MB89
0620	MB90
0630	MB91
0640	MB92
0650	MB93
0660	MB94
0670	MB95

Table 53-15. 16-byte message buffers

Address offset (hex)	MBDSR=b01 16-byte payload
0080	MB0
0098	MB1
00B0	MB2
00C8	MB3
00E0	MB4
00F8	MB5
0110	MB6
0128	MB7
0140	MB8
0158	MB9
0170	MB10
0188	MB11
01A0	MB12
01B8	MB13
01D0	MB14
01E8	MB15
0200	MB16
0218	MB17
0230	MB18
0248	MB19
0260	MB20
0280	MB21
0298	MB22
02B0	MB23
02C8	MB24
02E0	MB25
02F8	MB26
0310	MB27
0328	MB28
0340	MB29
0358	MB30
0370	MB31
0388	MB32
03A0	MB33
03B8	MB34
03D0	MB35
03E8	MB36
0400	MB37

Table continues on the next page...

Table 53-15. 16-byte message buffers (continued)

Address offset (hex)	MBDSR=b01 16-byte payload
0418	MB38
0430	MB39
0448	MB40
0460	MB41
0480	MB42
0498	MB43
04B0	MB44
04C8	MB45
04E0	MB46
04F8	MB47
0510	MB48
0528	MB49
0540	MB50
0558	MB51
0570	MB52
0588	MB53
05A0	MB54
05B8	MB55
05D0	MB56
05E8	MB57
0600	MB58
0618	MB59
0630	MB60
0648	MB61
0660	MB62

Table 53-16. 32-byte message buffers

Address offset (hex)	MBDSR=b10 32-byte payload
0080	MB0
00A8	MB1
00D0	MB2
00F8	MB3
0120	MB4
0148	MB5
0170	MB6
0198	MB7

Table continues on the next page...

Table 53-16. 32-byte message buffers (continued)

Address offset (hex)	MBDSR=b10 32-byte payload
01C0	MB8
01E8	MB9
0210	MB10
0238	MB11
0280	MB12
02A8	MB13
02D0	MB14
02F8	MB15
0320	MB16
0348	MB17
0370	MB18
0398	MB19
03C0	MB20
03E8	MB21
0410	MB22
0438	MB23
0480	MB24
04A8	MB25
04D0	MB26
04F8	MB27
0520	MB28
0548	MB29
0570	MB30
0598	MB31
05C0	MB32
05E8	MB33
0610	MB34
0638	MB35

Table 53-17. 64-byte message buffers

Address offset (hex)	MBDSR=b11 64-byte payload
0080	MB0
00C8	MB1
0110	MB2
0158	MB3
01A0	MB4

Table continues on the next page...

Table 53-17. 64-byte message buffers (continued)

Address offset (hex)	MBDSR=b11 64-byte payload
01E8	MB5
0230	MB6
0280	MB7
02C8	MB8
0310	MB9
0358	MB10
03A0	MB11
03E8	MB12
0430	MB13
0480	MB14
04C8	MB15
0510	MB16
0558	MB17
05A0	MB18
05E8	MB19
0630	MB20

53.4.37 Rx FIFO structure

When the CAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 53-18. Rx FIFO structure

	0 3 7 8	9	10	11	12 13 14 15	16 23	24 31
0x80		SRR	IDE	RTR	DLC	TIME STAMP	
0x84		ID standard				ID extended	
0x88	Data byte 0	Data byte 1				Data byte 2	Data byte 3
0x8C	Data byte 4	Data byte 5				Data byte 6	Data byte 7
0x90 to 0xDC	Reserved						
0xE0	ID filter table element 0						
0xE4	ID filter table element 1						
0xE8 to 0x2D4	ID filter table elements 2 to 125						
0x2D8	ID filter table element 126						
0x2DC	ID filter table element 127						

= Unimplemented or reserved

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the CAN_MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 53-19. ID table structure

	0	1	2 7	8 15	16	17	18 23	24 30	31
A	RTR	IDE	RXIDA (standard = 2–12, extended = 2–30)						
B	RTR	IDE	RXIDB_0 (standard = 2–12, extended = 2–15)		RTR	IDE	RXIDB_1 (standard = 18–28, extended = 18–31)		
C	RXIDC_0 (std/ext = 0–7)			RXIDC_1 (std/ext = 8–15)	RXIDC_2 (std/ext = 16–23)			RXIDC_3 (std/ext = 24–31)	

= Unimplemented or Reserved

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

53.5 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements.

For Classical CAN frames, simultaneous reception through FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes only. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it

possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 53-9). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 53-10).

The FlexCAN module is also able to receive and transmit messages in CAN FD format. The Message Buffers are sized to adequately store the quantity of data bytes selected by the MBDSRn bit fields in CAN_FDCTRL register. The quantity of FD MBs available for a given quantity of data bytes is described CAN_FDCTRL register. See also FlexCAN Memory Partition for CAN FD.

53.5.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the CAN_IFLAG register or by the interrupt request if enabled by the respective IMASK bit. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (CAN_MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB. When CAN_MCR[FDEN] is set, write also the EDL, BRS and ESI bits.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, FlexCAN adds the necessary number of bytes with constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, both CAN_CRC and CAN_FDCRC Registers are updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 53-9 and Table 53-10 in Message buffer structure).

When the Abort feature is enabled (CAN_MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked. Therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG bit before starting to prepare this MB for a new transmission or reception.

53.5.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CAN_CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CAN_CTRL1[LBUF] and CAN_MCR[LPRIOEN] bits settings.

53.5.2.1 Lowest-number Mailbox first

If CAN_CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. CAN_MCR[LPRIOEN] bit has no effect when CAN_CTRL1[LBUF] is asserted.

53.5.2.2 Highest-priority Mailbox first

If CAN_CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on CAN_MCR[LPRIOEN] bit setting.

53.5.2.2.1 Local Priority disabled

If CAN_MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 53-20. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

53.5.2.2.2 Local Priority enabled

If Local Priority is desired CAN_MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 53-21. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

53.5.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if the AEN bit in CAN_MCR register is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.

- During CAN BusOff state from TX_ERR_CNT=124 to 128. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

53.5.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox. No setup is required for EDL, BRS and ESI bits, they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 53-9 and Table 53-10 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.

4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should poll for frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 53-9 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. When CAN_MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx Mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when CAN_MCR[SRXDIS] bit is deasserted, FlexCAN can receive frames transmitted by itself if there exists a matching Rx Mailbox.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the BUF5I bit "Frames available in Rx FIFO" bit in the CAN_IFLAG1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional: needed only if a mask was used)
3. Read the Data field
4. Read the CAN_RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to CAN_IFLAG1[BUF5I] bit (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry)

53.5.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. The matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results are transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 53-22. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN

Table continues on the next page...

Table 53-22. Matching architecture (continued)

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the Rx SMB contents with the MB contents regardless the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY
- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the CAN_MCR[IRMQ] bit. If it is negated, the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CAN_CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless of the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found, then the matching winner determination is conditioned by the CAN_MCR[IRMQ] bit:
- - If CAN_MCR[IRMQ] bit is negated, the matching winner is the first matched Mailbox
  - If CAN_MCR[IRMQ] bit is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 53-23. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	

Table continues on the next page...

Table 53-23. Matching possibilities and resulting reception structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CAN_CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is

EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. See the description of the Rx Individual Mask Registers (CAN_RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care". Note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (CAN_RXFGMASK, CAN_RXMGMASK, CAN_RX14MASK and CAN_RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the CAN_MCR Register is negated.

53.5.5 Move process

There are two types of move process: move-in and move-out.

53.5.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:

    - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
    - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process
- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (CAN_MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined to the Rx FIFO.
2. Read all data words from the Rx SMB in accordance to the selected payload size for the Rx storage element.
3. Write all data words to the Rx Mailbox in accordance to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the Control/Status and ID words from the Rx SMB.
5. Write Control/Status and ID words to the Rx Mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

53.5.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

53.5.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

53.5.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- CAN_MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a Mailbox that is currently being transmitted or to a Mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode
- The module enters the BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.
- It is necessary to reconfigure the EDL, BRS, and ESI bits of the aborted MB before transmitting it again.

53.5.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instructions on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without setting the respective IFLAG

In order to perform a *safe inactivation* and avoid the above consequences for Tx Mailboxes, the CPU must use the Transmission Abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

53.5.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code different than INACTIVE

(0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the Rx SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (CAN_TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation), and it takes place only when the module resumes to Normal or Freeze modes.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

53.5.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CAN_MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature.

CAUTION

Rx FIFO must not be enabled when CAN FD feature is enabled.

The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the CAN_RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the CAN_RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the CAN_IFLAG1[BUF5I] is asserted.

The CAN_IFLAG1[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The CAN_IFLAG1[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CAN_CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the CAN_RXFIR register. The CAN_RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the CAN_IFLAG1[BUF5I] flag is asserted. The CAN_RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (CAN_RXIMRx), according to the setting of CAN_CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the CAN_MCR[IRMQ] bit is negated, then the FIFO filter table is affected by CAN_RXFGMASK.

53.5.8 CAN protocol related features

This section describes the CAN protocol related features.

53.5.8.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure detection capability that was in the original CAN FD protocol, which is also called non-ISO CAN FD by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-FD CAN FD protocol. The non-ISO CAN FD is still supported by FlexCAN so that it can be used mainly during an intermediate phase, for evaluation and development purposes.

Therefore, it is strongly recommended to configure FlexCAN to the ISO CAN FD protocol by setting the ISOCANFDEN field in the CAN_CTRL2 register.

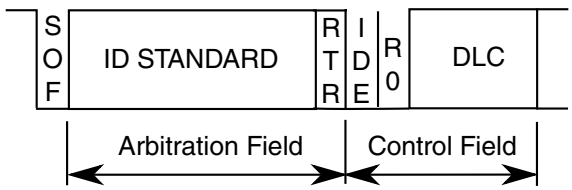
53.5.8.2 CAN FD frames

The ISO 11898-1 standard specifies the Classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN Flexible Data Rate Frame format. The Classical Frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The Flexible Data Rate Frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

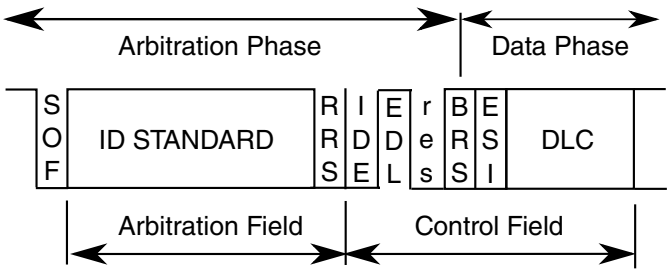
There are three additional control bits in the CAN FD frame. The Extended Data Length (EDL) bit enables a longer data payload with different data length coding. The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame. The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes. There is no Remote Frames (see Remote frames) in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in the Classical CAN format. When a FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is negated. The RTR bit must be considered in classical frames only.

CAN FD messages may be formatted as long frames where the data field exceeds 8 bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame. Messages in Classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. The following figure illustrates the message formats for Classical and FD frames with either standard or extended ID.

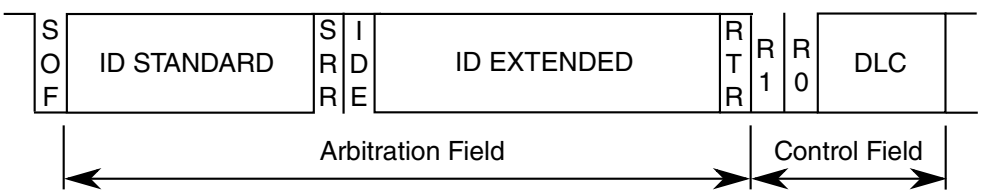
CAN Standard Format



CAN FD Standard Format



CAN Extended Format



CAN FD Extended Format

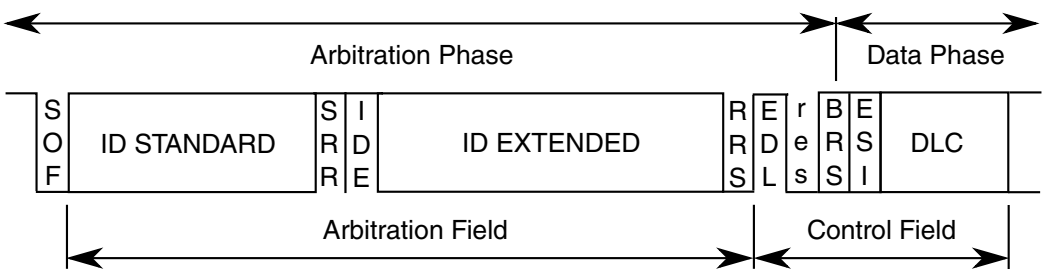


Figure 53-2. CAN message formats

The ability to receive and transmit CAN FD messages is enabled by the CAN_MCR[FDEN] bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, while a

Classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded in accordance to the DLC field value (see DLC definition in Message buffer structure).

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start Of Frame (SOF) bit until the Bit Rate Switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, also named the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values. Finally, from the CRC Delimiter until the Intermission bits, the transmission resumes to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the CAN_CBT register (also by CAN_CTRL1 register for backward compatibility). Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the CAN_FDCBT register.

NOTE

If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and will last until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing in order to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use FD frames.

CAN_FDCTRL[FDRATE] enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. If FDRATE is negated, the transmission is performed at nominal rate regardless of the BRS bit value. The CAN_FDCTRL[FDRATE] bit can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. The following figure describes the mechanism for entering and leaving the data phase when BRS bit is recessive.

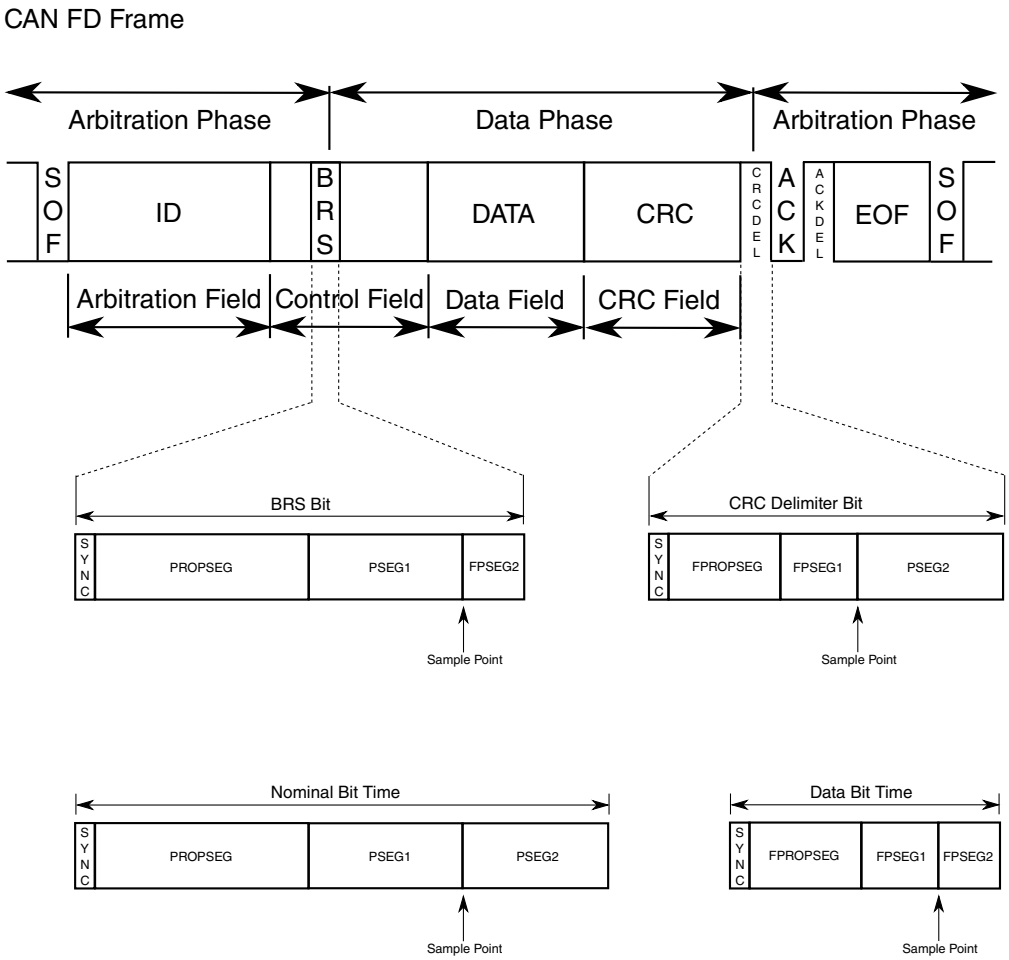


Figure 53-3. Bit rate switching mechanism for CAN FD messages

NOTE

In Classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. FlexCAN sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of CAN_PE sub-block. For example, with a CAN_PE clock frequency of 40MHz and the shortest configurable bit time of 8 time quanta, the bit rate in the data phase is 5 Mbit/s.

The value of the ESI bit is determined either by the transmitter's error state at the start of the transmission, if the frame is originated in the FlexCAN node, or by the original transmitting node in case FlexCAN is acting as a gateway for the message. If the transmitter is error passive, ESI is transmitted recessive; otherwise, it is transmitted dominant.

There are different CRC polynomials for different CAN frame formats. The first polynomial, CRC_15, is used for all frames in Classical CAN format. The second, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. Each polynomial results in a Hamming Distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, FlexCAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CAN_FDCRC register stores the calculated CRC for the transmitted message, with the adequate length in accordance to the type of message, for both CAN FD and non-FD messages. The CAN_CRCR register reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When FlexCAN is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A Stuff Error is detected if the fixed stuff bit has the same value as its preceding bit.

FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters RXERRCNT and TXERRCNT in the CAN_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indistinctly. There are two extra error counters (RXERRCNT_FAST and TXERRCNT_FAST) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see CAN_ECR register).

Error Flags BITERR1, BITERR0, ACKERR, CRCERR, FRMERR and STFERR in the ESR1 register report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if CAN_CTRL1[ERRMSK] is asserted. The CAN_ESR1 register has

additional error flags (BITERR1_FAST, BITERR0_FAST, CRCERR_FAST, FRMERR_FAST and STFERR_FAST) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported in CAN_ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames, and is based on RXERRCNT and TXERRCNT error counters only. Information contained in RXERRCNT_FAST and TXERRCNT_FAST counters may be considered as status to help detect the error nature related to the bit rate value.

When FlexCAN is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an Error Flag.

Resynchronization and Hard Synchronization occur in CAN FD frames in the same way as in Classical CAN ones. Additionally, a Hard Synchronization is also performed at the recessive to dominant edge from EDL to R0 in CAN FD format frames. FlexCAN does not resynchronize while transmitting in the CAN FD data phase.

53.5.8.3 Transceiver Delay Compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the Transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

FlexCAN supports an optional Transceiver Delay Compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit in order to check for bit errors.

The TDC mechanism can be enabled by the CAN_FDCTRL[TDCEN] bit and is effective only during the data phase of FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE

The actual value of the CRC Delimiter bit is disregarded by transmitters using the Transceiver Delay Compensation mechanism. A global error at the end of the CRC Field will cause the receivers to send error frames that the transmitter will detect during Acknowledge or End of Frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit (as shown in the next figure). The loop delay is measured in Protocol Engine (PE) clock periods (CANCLK, see Protocol timing), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in CAN_FDCTRL[TDCOFF]. CAN_FDCTRL[TDCVAL] bit field stores the result of this calculation. The TDCVAL value saturates at its maximum value of 15 CANCLK when the delay measurement is too long.

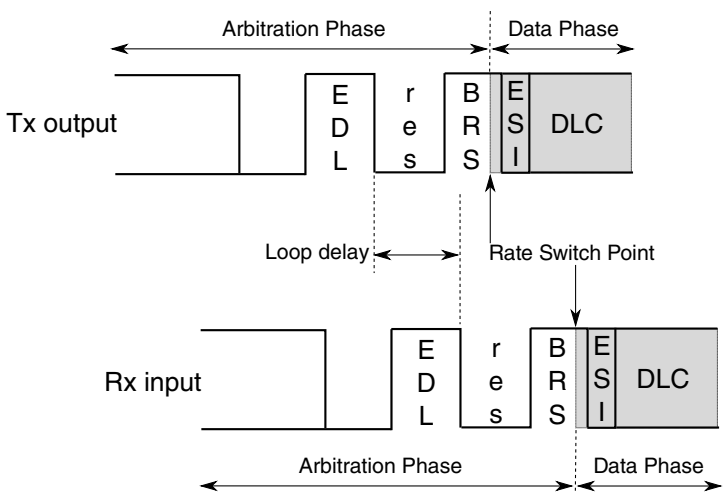


Figure 53-4. Transceiver loop delay measurement

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time (e.g. half of the bit time in the data phase), far away from its edges. Therefore, the TDCOFF value cannot be larger than the CAN bit duration in the data phase.

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on TDCVAL is started which ends with the comparison of the

received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase the delay compensation is always disabled. The maximum delay which can be compensated by the FlexCAN's transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the CAN_FDCTRL[TDCFAIL] flag is set to indicate when the Transceiver Delay Compensation mechanism is out of range, unable to compensate the transceiver loop delay.

53.5.8.4 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by configuring the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for

filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

NOTE

There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant RRS bit. FlexCAN receives and transmits remote frames in the Classical CAN format.

53.5.8.5 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

53.5.8.6 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The Free Running Timer is not incremented during Disable, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in Control 1 Register (CAN_CTRL1).

53.5.8.7 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CAN_CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock. In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (MDIS bit set in the Module Configuration Register).

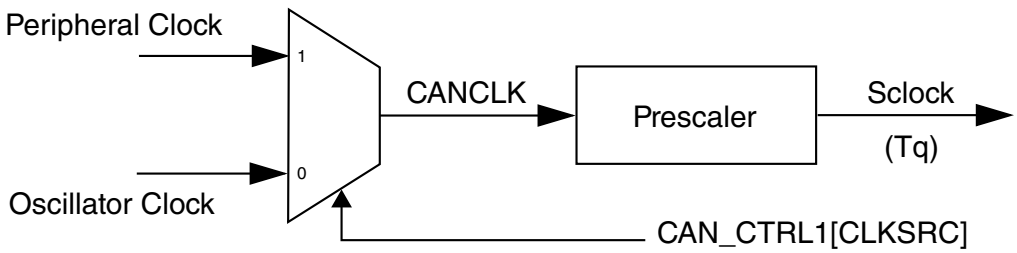


Figure 53-5. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than the peripheral clock.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1. The CAN FD Bit Timing register (CAN_FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

NOTE

When the CAN FD feature is enabled, always set CAN_CBT[BTF] and configure the CAN bit timing variables in CAN_CBT. See CAN Bit Timing Register (CAN_CBT).

The PRESDIV field (as well as its extended range EPRESDIV and FDPRESDIV for the data phase bits of CAN FD messages) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{CANCLK}}$$

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

A bit time is subdivided into three segments[1] (see Figure 53-6, Figure 53-7 and Table 53-24):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CAN_CTRL1 Register so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPROPSEG and EPSEG1 fields from CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPROPSEG and FDPSEG1 from CAN_FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CAN_CTRL1 Register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPSEG2 fields of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPSEG2 from CAN_FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The Time Segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FlexCAN.

NOTE

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an

1. For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

Information Processing Time (IPT) of 2, which is the value implemented in the FlexCAN module.

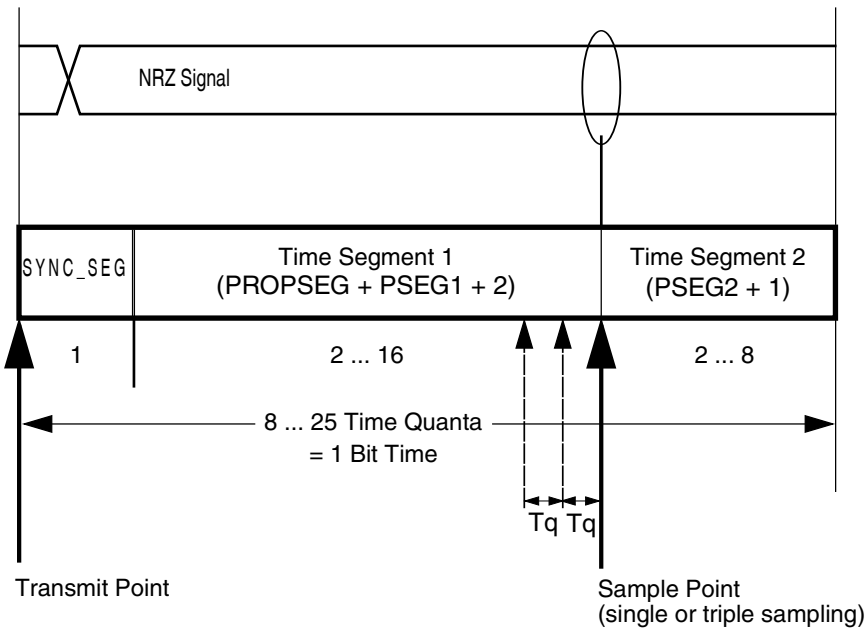


Figure 53-6. Segments within the bit time (example using CAN_CTRL1 bit timing variables for Classical CAN format)

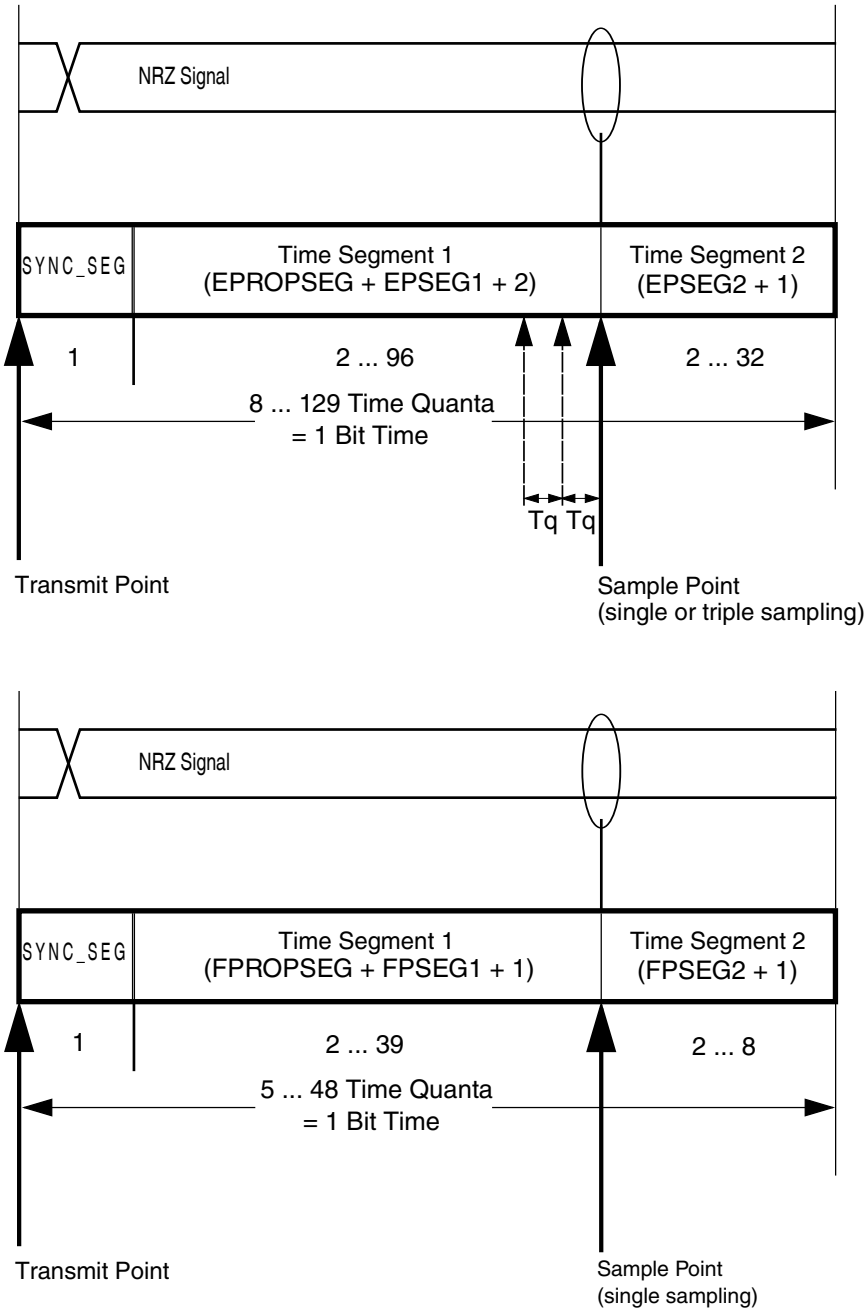


Figure 53-7. Segments within the bit time (example using CAN_CBT and CAN_FDCBT bit timing variables for CAN FD format)

Table 53-24. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.

Table continues on the next page...

Table 53-24. Time segment syntax (continued)

Syntax	Description
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 53-25. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

Whenever CAN bit is used as a measure of time duration (e.g. estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$\text{NumClkBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

where:

- NumClkBit is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CAN_CTRL1[PSEG1] field;
- PSEG2 is the value in CAN_CTRL1[PSEG2] field;
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field;
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing Register (CAN_CBT) and also to the CAN FD Bit Timing Register (CAN_FDCBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

53.5.8.8 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

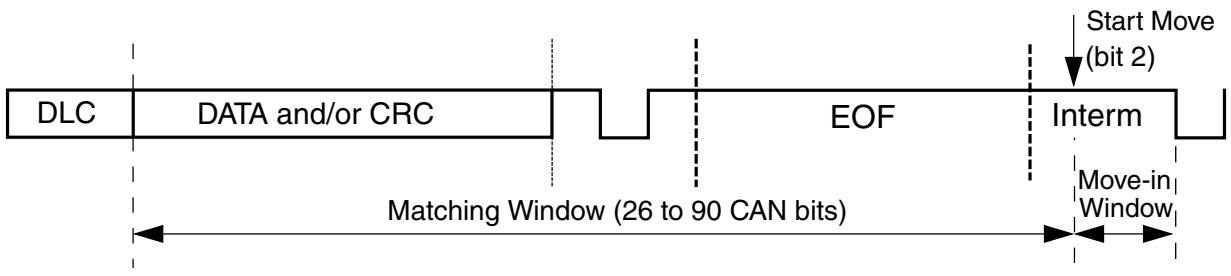


Figure 53-8. Matching and move-in time windows

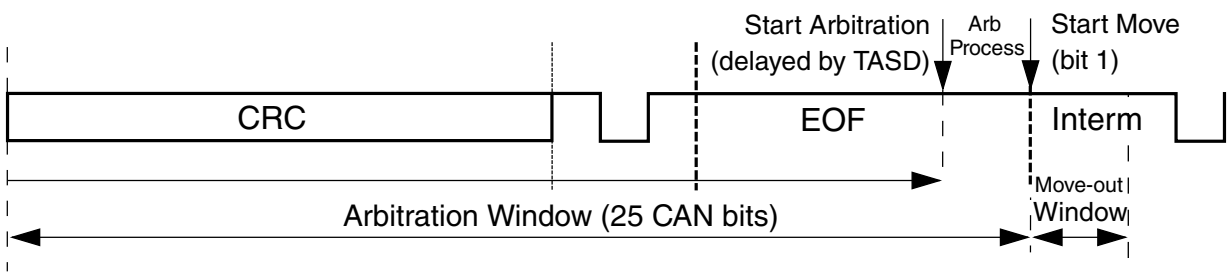


Figure 53-9. Arbitration and move-out time windows

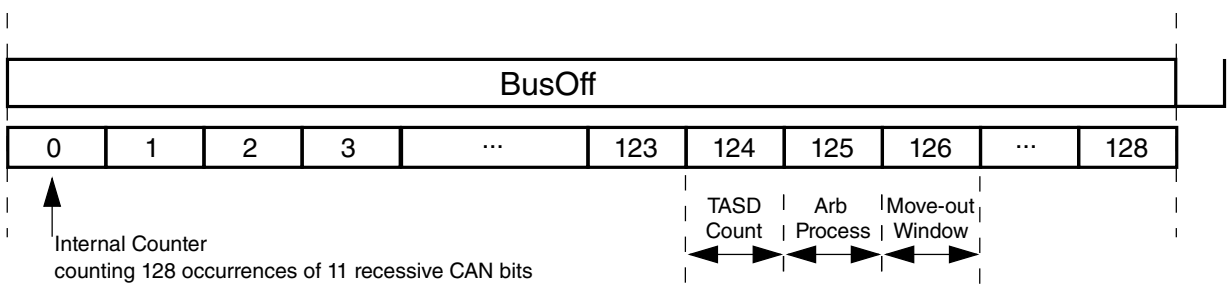


Figure 53-10. Arbitration at the end of bus off and move-out time windows

NOTE

In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent

memory access due to the CPU or other internal FlexCAN sub-blocks.

53.5.8.9 Tx Arbitration start delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2[TASD]) is a variable that indicates the number of CAN bits used by FlexCAN to delay the Tx Arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure Message Buffers (MBs) for transmission after the end of the internal Arbitration process, where FlexCAN finds the winner MB for transmission (see Arbitration process). If the Arbitration ends too early before the first bit of Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the Arbitration start point, as shown in the next figure, based on factors such as:

- The peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- The number of Message Buffers (MBs) in use by the Matching and Arbitration processes.

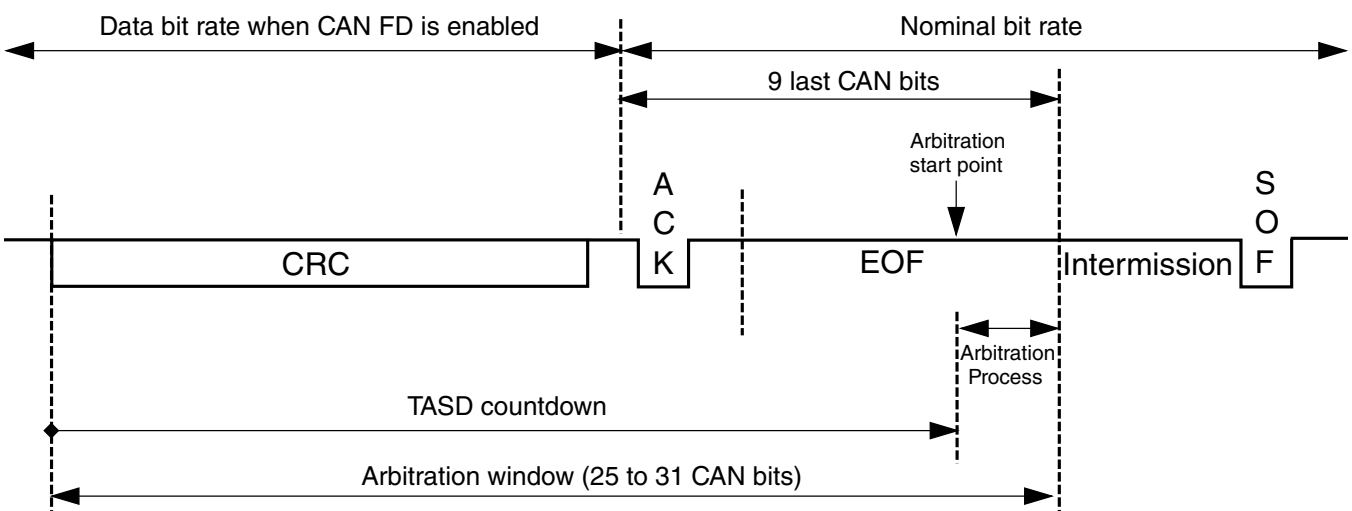


Figure 53-11. Optimal Tx Arbitration start point

The duration of an Arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal Arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the Arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0 then the Arbitration start is not delayed and more time is reserved for Arbitration. On the other hand, if TASD is close to 24 then the CPU can configure a Tx MB later and less time is reserved for Arbitration. If too little time is reserved for Arbitration the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and $(MAXMB + 1) \leqslant NMB_{END}$

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCB_N}$$

For CAN FD frames and $(MAXMB + 1) > NMB_{END}$

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

where:

$$NMB_{END} = \frac{(9 * CPCB_N) - 4}{2}$$

$$BITRATE_N = \left(\frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] \times (EPRESDIV + 1)} \right)$$

$$BITRATE_F = \left(\frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] \times (FPRESDIV + 1)} \right)$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

- MAXMB is the value in CAN_CTRL1[MAXMB] field
- NMB_{END} is the number of Message Buffers that can be scanned by the Arbitration process during the 9 last CAN bits at the end of a frame, see the figure above
- $BITRATE_N$ is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables
- $BITRATE_F$ is the CAN bit rate in bits per second calculated by the data CAN bit time variables
- $CPCB_N$ is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames
- $CPCB_F$ is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames
- f_{CANCLK} is the oscillator clock, in Hz
- f_{SYS} is the peripheral clock, in Hz
- EPSEG1 is the value in CAN_CBT[EPSEG1] field (CAN_CTRL1[PSEG1] can also be used)
- EPSEG2 is the value in CAN_CBT[EPSEG2] field (CAN_CTRL1[PSEG2] can also be used)
- EPROPSEG is the value in CAN_CBT[EPROPSEG] field (CAN_CTRL1[PROPSEG] can also be used)

- EPRESDIV is the value in CAN_CBT[EPRESDIV] field (CAN_CTRL1[PRESDIV] can also be used)
- FPSEG1 is the value in CAN_FDCBT[FPSEG1] field
- FPSEG2 is the value in CAN_FDCBT[FPSEG2] field
- FPROPSEG is the value in CAN_FDCBT[FPROPSEG] field
- FPRESDIV is the value in CAN_FDCBT[FPRESDIV] field

See also Protocol timing for more details.

The following tables give the TASD value calculated for some configuration cases.

Case 1:

- Clock ratio = 2:1 (example: peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 53-26. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	24	8.0
64	23	8.0
96	22	8.0

Case 2:

- Clock ratio = 1:1 (example: peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 53-27. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	6.67
54	22	5.0
64	21	3.33
96	20	1.6

Case 3:

- Clock ratio = 2:1 (example: peripheral clock 40 MHz and oscillator clock 20 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 53-28. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	4.0
54	22	4.0
64	21	3.33
96	20	1.54

53.5.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency can not be smaller than the oscillator clock frequency
- There must be a minimum number of peripheral clocks per CAN bit, as specified in the table shown below

Table 53-29. Minimum number of peripheral clocks per CAN bit for Classical CAN format

Number of Mailboxes	Value of CAN_MCR[RFEN] bit	Minimum number of peripheral clocks per CAN bit
16	0	16

Table continues on the next page...

Table 53-29. Minimum number of peripheral clocks per CAN bit for Classical CAN format (continued)

Number of Mailboxes	Value of CAN_MCR[RFEN] bit	Minimum number of peripheral clocks per CAN bit
32	0	16
64	0	25
96	0	37
16	1	16
32	1	17
64	1	30
96	1	42

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in the preceding table determines the minimum peripheral clock frequency for a given number of Mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values contained in either the Control 1 Register (CAN_CTRL1) or CAN Bit Time register (CAN_CBT). The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be 8; therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

For CAN FD frame format, there are some constraints that need to be satisfied. The number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit) can be calculated by the equation below.

$$\text{NumClkNomBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

$$= \frac{f_{SYS}}{\text{NomBitRate}}$$

where PRESDIV, PSEG1 and PSEG2 are CAN bit time values in CTRL1 register. Alternatively, EPRESDIV, EPSEG1 and EPSEG2 values in CBT register can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the Arbitration Phase (NomBitRate) as shown in the equation above.

The number of CAN bits in the Data Phase of a FD Frames with the BRS bit set (fast CAN bits, in short) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined in the table below. The less the

number of data bytes, the less the number of fast CAN bits, and less time is available for FlexCAN to scan the whole Message Buffer memory during the internal matching and arbitration processes.

Table 53-30. Number of fast CAN bits in a CAN FD frame

Minimum number of data bytes	DLC field	NumOfFastBits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the Data Phase, where the CAN bit rate is faster than in the Arbitration Phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for FlexCAN to scan the Message Buffer memory during reception and transmission. The equation below calculates this constraint.

$$MinNumClkFastBit_A = \frac{(8.5 \times MaxNumOfMb) + 64 - (9 \times NumClkNomBit)}{NumOfFastBits}$$

where MaxNumOfMb is the maximum number of available Mailboxes defined in CAN_MCR[MAXMB].

The clock domain crossing circuit between the CHI and PE sub-blocks also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without loosing status information through the interface, as shown in the equation below.

$$MinNumClkFastBit_B = 3 \times \left\langle 1 + \frac{f_{SYS}}{f_{CANCLK}} \right\rangle$$

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above.

$$MinNumClkFastBit = Maximum\ (\ MinNumClkFastBit_A, MinNumClkFastBit_B)$$

Then, the maximum CAN bit rate in the Data Phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$DataBitRate_{MAX} = \frac{f_{CANCLK}}{ROUNDUP\left\langle \frac{MinNumClkFastBit \times f_{CANCLK}}{f_{SYS}} \right\rangle}$$

The peripheral and oscillator clock frequencies, the maximum number of mailboxes and the expected nominal bit rate affect the maximum data bit rate attainable by FlexCAN in CAN FD mode. Besides, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the CAN FD bit rate is affected by the configuration of FlexCAN variables, an application example with the peripheral and oscillator clock frequencies set to 50 MHz and 40 MHz, respectively, is considered.

Step 1 - Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as below.

$$NumClkNomBit = \frac{50 \times 10^6}{1 \times 10^6} = 50$$

Step 2 - The number of fast CAN bits (NumOfFastBits) is determined in the table presented above. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the Data Phase.

Step 3 - Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

$$\text{MinNumClkFastBit}_A = \frac{(8.5 \text{ X } 96) + 64 - (9 \text{ X } 50)}{85} = 5.06$$

$$\text{MinNumClkFastBit}_B = 3 \text{ x } \left(1 + \frac{50}{40} \right) = 6.75$$

$$\text{MinNumClkFastBit} = \text{Maximum } (5.06, 6.75) = 6.75$$

Step 4 - The maximum CAN bit rate in the Data Phase can be finally found.

$$\text{DataBitRate}_{MAX} = \frac{40 \text{ x } 10^6}{\text{ROUNDUP} \left(\frac{6.75 \text{ x } 40 \text{ x } 10^6}{50 \text{ x } 10^6} \right)} = 6.667 \text{ Mbps}$$

As demonstrated in this example, even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbps. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

The table below shows the maximum data rate for CAN FD according to clock frequencies, payload size and number of available mailboxes. See in this table that, for some cases, if the number of available mailboxes is reduced, the FlexCAN can then achieve a data rate up to 8 Mbps.

Table 53-31. Maximum CAN bit rate in Data Phase on CAN FD frames

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
40	8	94	6.667
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714

Table continues on the next page...

Table 53-31. Maximum CAN bit rate in Data Phase on CAN FD frames (continued)

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
50	12 to 64	128	6.667
60	8	126	8.0
60	12	128	8.0
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

53.5.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

"Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

53.5.10.1 Freeze mode

This mode is requested either by the CPU through the assertion of the HALT bit in the CAN_MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the CAN_MCR Register and the module is not in a low-power mode. This mode is also requested by FlexCAN through the automatic assertion of both HALT and FRZ bits when CAN_MECR[NCEFAFRZ] bit is set and a non-correctable error is detected in a memory read access performed by FlexCAN internal processes (see Response to Errors).

The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state

- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in does not prevent going to Freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in CAN_MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in CAN_MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CAN_CTRL1[CLKSRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the CAN_MCR Register
- The chip is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

53.5.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the CAN_MCR[MDIS] bit, and the acknowledgement is obtained through the assertion by the FlexCAN of the CAN_MCR[LPMACK] bit. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPMACK bit and negates the FRZACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.

- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in CAN_MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPMACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

53.5.10.3 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in CAN_MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus.

53.5.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CAN_IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CAN_MCR[RFEN] = 1), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 24 of the CAN_IFLAG1 register becomes the "FIFO Overflow" flag; bit 25 becomes the "FIFO Warning" flag, bit 26 becomes the "Frames Available in FIFO" flag and bits 27-30 are unused. See the description of the Interrupt Flags 1 Register (CAN_IFLAG1) for more information.

CAUTION

FIFO cannot be enabled when CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Error, Error Fast, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 Register.

53.5.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- It is possible for the RXIMR memory region to be considered as general purpose memory and available for access. There are two ways of doing this:
  a. If CAN_MCR[IRMQ] is cleared, the individual masks (RXIMR) are disabled. In this case the RXIMR memory region is considered as general purpose memory.
  b. If CAN_MCR[MAXMB] is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN's RAM can support up to 16 MBs, CAN_CTRL2[RFFN] is 0x0, and CAN_MCR[MAXMB] is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

53.5.13 Detection and Correction of Memory Errors

FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits that ensure a Hamming distance of 4. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

FlexCAN detects a non-correctable error in the event either an all-zeros or an all-ones read occurs. See CAN_RERRSYNR register description.

Memory errors are indicated to the Host through status register (Error Status Register (CAN_ERRSR)) and bus transfer errors, and reported through report registers (Error Report Address Register (CAN_RERRAR), Error Report Data Register (CAN_RERRDR) and Error Report Syndrome Register (CAN_RERRSYNR)).

The error detection and correction mechanism can be activated or not, controlled by the ECCDIS bit in Memory Error Control Register (CAN_MECR). When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated to the data.

NOTE

All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. The WRMFRZ bit in Control 2 Register (CAN_CTRL2) grants write access to all memory positions that require initialization, ranging from 0x080 to 0xADF and from 0xF28 to 0xFFF when the CAN FD feature is enabled. The RXMGMASK, RX14MASK, RX15MASK, and RXFGMASK registers need to be initialized as well. The CAN_MCR[RFEN] bit must not be set during memory initialization.

To avoid accidentally changing the critical error correction configuration, this protocol must be followed to enable the update of the Memory Error Control Register (CAN_MECR):

1. By default, ECRWRE bit in Control 2 Register (CAN_CTRL2) is 0 and ECRWRDIS bit in Memory Error Control Register (CAN_MECR) is 1.
2. Set ECRWRE bit in Control 2 Register (CAN_CTRL2).
3. Clear ECRWRDIS bit in Memory Error Control Register (CAN_MECR).

4. All writes to Memory Error Control Register (CAN_MECR) must keep ECRWRDIS cleared.
5. After configuration is done, lock the Memory Error Control Register (CAN_MECR) by either setting ECRWRDIS or clearing ECRWRE.

53.5.13.1 Sources of the Memory Access

The FlexCAN memory can be accessed by two major sources (or requestors):

- by Host (CPU): the largest word accessed is 32-bit wide
- by FlexCAN internals processes (Rx Matching, Tx Arbitration, Move-in on reception, Move-out on transmission): the largest word accessed is 64-bit wide

The way that non-correctable errors are indicated and reported depends on the source of access.

53.5.13.2 Error Indication

Memory errors are indicated by flags HANCEIF, FANCEIF, CEIF in the Error Status Register (CAN_ERRSR). Non-correctable errors detected in memory reads requested by Host are indicated separately than the ones detected in requests by FlexCAN internal processes. FlexCAN makes no distinction of the source of the access when correctable errors are detected. There are 3 independent flags for these 3 cases. If both non-correctable and correctable errors are found in different bytes in the same read operation, both flags are set.

A non-correctable error detected in Host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked.

Each indication flag has one Overrun flag in Error Status Register (CAN_ERRSR). The Overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. This is the recommended handling sequence for error indication:

1. Get error report information from report registers
2. Use this information to take proper measures in the application
3. Clear the HANCEIF, FANCEIF, CEIF flags
4. If the Overrun flag is active:
    a. Alert application that at least one error could not be handled

b. Clear the Overrun flag

The FlexCAN internal processes can access memory in transactions larger than 32-bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words the Interrupt and Overrun flags are set simultaneously.

53.5.13.3 Error Reporting

The report registers Error Report Address Register (CAN_RERRAR), Error Report Data Register (CAN_RERRDR) and Error Report Syndrome Register (CAN_RERRSYNR) provide detailed information about the address read, raw data and syndrome read with error and indicated by the flags described in Error Indication. The address, data and syndrome registers are updated simultaneously along with the error flags, according to these rules:

1. If any of the 2 non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved)
2. Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error; or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected

Reporting of errors detected in accesses larger than 32-bit follows the rules described in Error Indication and in the Error Report Address Register (CAN_RERRAR) description.

The address reported in CAN_RERRAR and defined in CAN_ERRIAR are not the same listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the Error Injection Address Register (CAN_ERRIAR) description.

Addresses reported when reading memory portions organized as FIFOs, such as the Rx FIFO Structure and the Rx FIFO Information Register (CAN_RXFIR), refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

To assure coherence of the error report registers it is necessary to turn off the report update by setting the CAN_MECR[RERRDIS] bit before reading the report registers.

53.5.13.4 Response to Errors

Correctable errors have no consequence on FlexCAN operation because affected data is corrected before its use by the host or FlexCAN internal processes.

For host-initiated reads, a non-correctable error may affect the host, but does not affect FlexCAN operation.

Non-correctable errors detected on memory reads requested by the FlexCAN internal processes may result in incorrect operation depending on the state of the NCEFAFRZ bit in CAN_MECR register, as follows:

- During reception (either Matching or Move-in processes), when a non-correctable error occurs, an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode to prevent corrupted data from being treated as valid by FlexCAN internal processes. When NCEFAFRZ is negated, FlexCAN continues working and a corrupted frame is received.
- During Arbitration process, when a non-correctable error occurs, either a non-highest priority Tx Message Buffer may be mistakenly selected for transmission or its data may be corrupted. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode before starting the Move-out. When NCEFAFRZ is negated, FlexCAN proceeds to Move-out with a corrupted frame that will be transmitted on the CAN bus.
- During Move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the Arbitration to the Tx SMB for transmission. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode before starting the transmission. When NCEFAFRZ is negated, the corrupted frame is transferred from the Tx SMB to the Protocol Engine (PE) sub-block and is transmitted on the CAN bus.
- A non-correctable error can also be detected beyond the Move-out process, when Tx data is read from Tx SMB (buffer located in RAM) to be transferred to the PE sub-block for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the CAN bus. To prevent the frame from being successfully received by the external nodes, FlexCAN inverts all bits in the CRC field (CRC sequence plus CRC Delimiter), and transmits an Error Flag just after CRC Delimiter as a result of self-detecting a Bit1 Error and a Form Error due to the CRC field inversion. When NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode just after the Error Frame. When NCEFAFRZ is negated, FlexCAN may attempt to re-transmit the same frame, as long as no other higher priority Tx MB is subsequently configured for transmission. In the event the non-correctable error persists, FlexCAN eventually reaches the Bus Off state because of consecutive error detections. The CAN_ECR[TXERRCNT] is updated every time the FlexCAN inverts the CRC field causing errors as described above.

When NCEFAFRZ is set and FlexCAN enters in Freeze Mode, only the CPU can cause FlexCAN to exit Freeze mode and resume Normal Mode. The assertion of the CAN_MECR[NCEFAFRZ] bit is the only way to prevent corrupted frames from being transmitted on the CAN bus up to the Move-out internal process.

The error report registers can provide information to the application for a customized handling of these situations.

53.5.13.5 Error Injection

The error injection registers CAN_ERRIAR, CAN_ERRIDPR, and CAN_ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. The relation between the error injection addresses and the respective ones in the module memory map is shown in the CAN_ERRIAR description.

The injection is done by flipping the data and parity bits correspondent to the bits in 1 in CAN_ERRIDPR and CAN_ERRIPPR. Injection can be selected specifically for memory accesses requested by Host or by FlexCAN internal processes.

In case of accesses larger than 32-bits, the EXTERRIE bit in Memory Error Control Register (CAN_MECR) extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE

It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRIE and FAERRIE bits in Memory Error Control Register (CAN_MECR) must be cleared while configuring the memory injection registers.

53.6 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

53.6.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- Chip level hard reset, which resets all memory mapped registers asynchronously

- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 53-4 to see what registers are affected by soft reset.
- Chip level soft reset, which has the same effect as the SOFTRST bit in MCR

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The CAN_MCR[SOFTRST] bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source should be selected while the module is in Disable mode (see CAN_CTRL1[CLKSRC] bit). After the clock source is selected and the module is enabled (CAN_MCR[MDIS] bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in CAN_MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the CAN_MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode (see Freeze mode). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register (CAN_MCR)
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRNEN bit
  - If required, disable frame self reception by setting the SRXDIS bit
  - Enable the Rx FIFO by setting the RFEN bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control 1 Register (CAN_CTRL1) and optionally the CAN Bit Timing Register (CAN_CBT). Initialize also the CAN FD CAN Bit Timing Register (CAN_FDCBT).
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW

    - Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, ERJW
    - Determine the CAN FD bit timing parameters: FPROPSEG, FPSEG1, FPSEG2, FRJW
    - Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field
    - Determine the CAN FD bit rate by programming the FPRESDIV field
    - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
    - The Control and Status word of all Message Buffers must be initialized
    - If Rx FIFO was enabled, the ID filter table must be initialized
    - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers (CAN_RXIMRn)
- Set required interrupt mask bits in the CAN_IMASK Registers (for all MB interrupts) and in CAN_CTRL1 / CAN_CTRL2 Registers (for Bus Off and Error interrupts)
- Negate the HALT bit in CAN_MCR

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

Chapter 54
Zipwire

54.1 Chip-specific Zipwire information

This section summarizes the module configuration in the controller. For a comprehensive description of SIPI and LFAST, please refer the specific dedicated chapters for SIPI and LFAST.

The SIPI and LFAST are connected to one another and appear as a single unit. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules (see Figure 54-1). Zipwire is official brand for the combination of LFAST and SIPI.

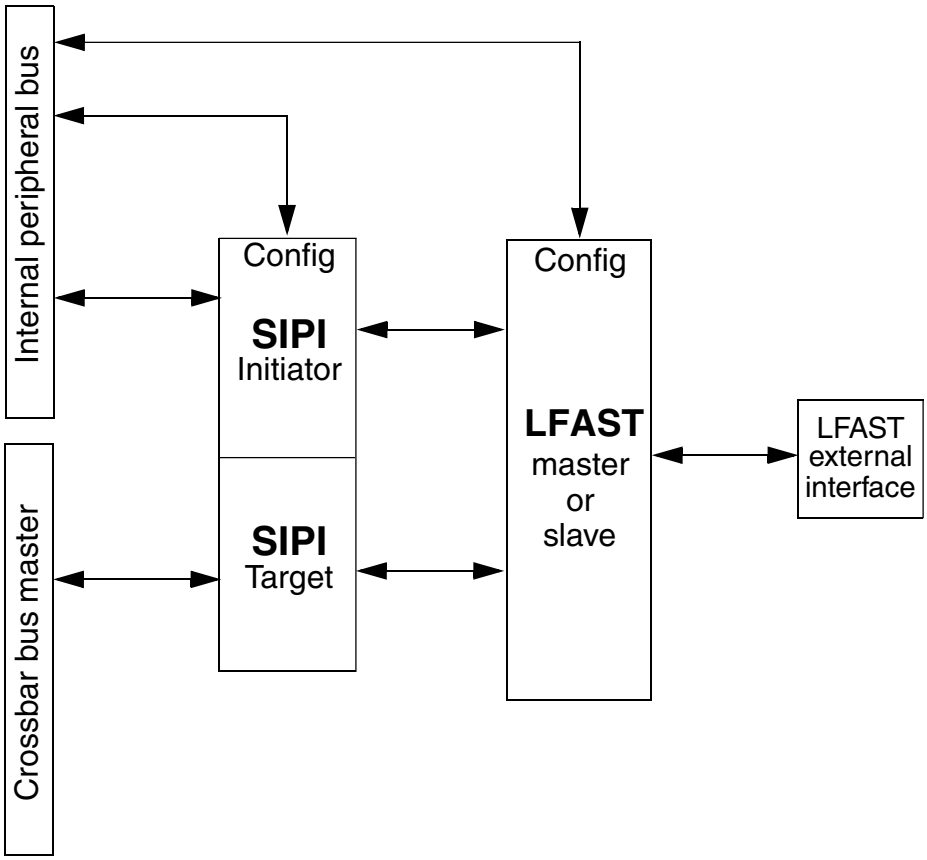


Figure 54-1. LFAST and SIPI block diagram

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality. The SIPI CHIP ID = 098D801D.

54.2 Overview

The SIPI and LFAST modules work together as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

54.3 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals and/or memories of another MCU or smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the intitiator or target node. Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and support multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at a high speed using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate LFAST reference clock for a total of five pins.

Zipwire also supports a streaming mode for the transfer of large blocks of data.

In streaming mode Zipwire has a high transfer rate.

54.4 Zipwire Block Diagram

Figure 54-2 shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target mode transactions. The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST reference clock coming from MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

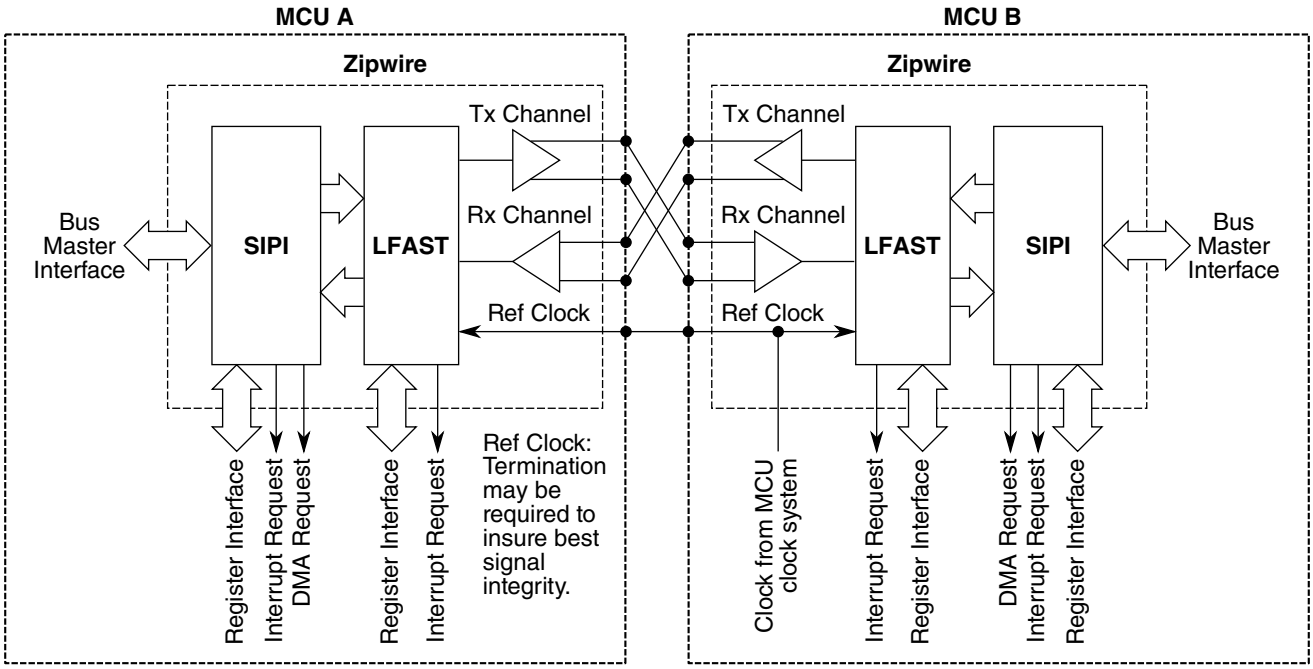


Figure 54-2. Zipwire connection diagram

54.5 Architecture

Zipwire is composed of several modules and system resources. The physical layer is LVDS with common mode voltage and defined swing.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). SIPI runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload in Zipwire:

- 32-bit
- 64-bit

- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated by one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently of each other and can run simultaneously. The SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Inititator MCU and executed by DMA and SIPI without CPU intervention at either Initiator or Target MCU.

54.6 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx configuration is controlled by LFAST Control registers .
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Directly connected to the SIPI module. Allows received data to be efficiently transferred to SIPI and transmit data to be transferred from SIPI.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers.

- DMA connections — Allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

54.7 Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

54.7.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.
- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in a LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is *c* Mbaud which yields a bit time of *d* ns

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 89*d* ns.

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of $n \times$ XBAR cycles to read different memories.
- XBAR clocked at *x* MHz
- SIPI clocked at *y* MHz

Therefore, time for SIPI to read an address location is:

$(13 \times y \text{ MHz clocks}) + (n \times x \text{ MHz clocks}) = z \text{ ns}$

54.7.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = 4 bytes in z ns = 4 × z MB/second

54.7.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 89d ns + z ns + 89d ns = z + 178d ns

54.7.2 Write performance

A Zipwire write operation consists of three stages:

- Initiator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address
- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header

- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is *c* Mbaud which yields a bit time of *d* ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 121*d* ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 57*d* ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of $m \times$ XBAR cycles to write different memories.
- XBAR clocked at *x* MHz
- SIPI clocked at *y* MHz

Therefore, the time for SIPI to write an address location is:

$(13 \times y \text{ MHz clocks}) + (m \times x \text{ MHz clocks}) = v \text{ ns}$

54.7.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in v ns = 4 × v MB/second

54.7.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 121d ns + v ns + 57d ns = v + 178d ns

Chapter 55
Serial Interprocessor Interface (SIPI)

55.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

55.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between 2 devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface

- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

55.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

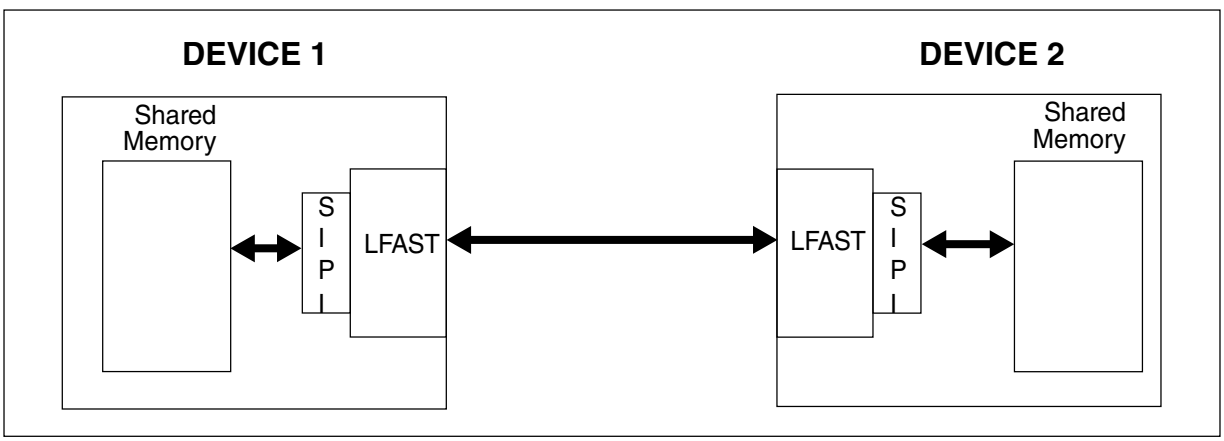


Figure 55-1. Interprocessor communication diagram

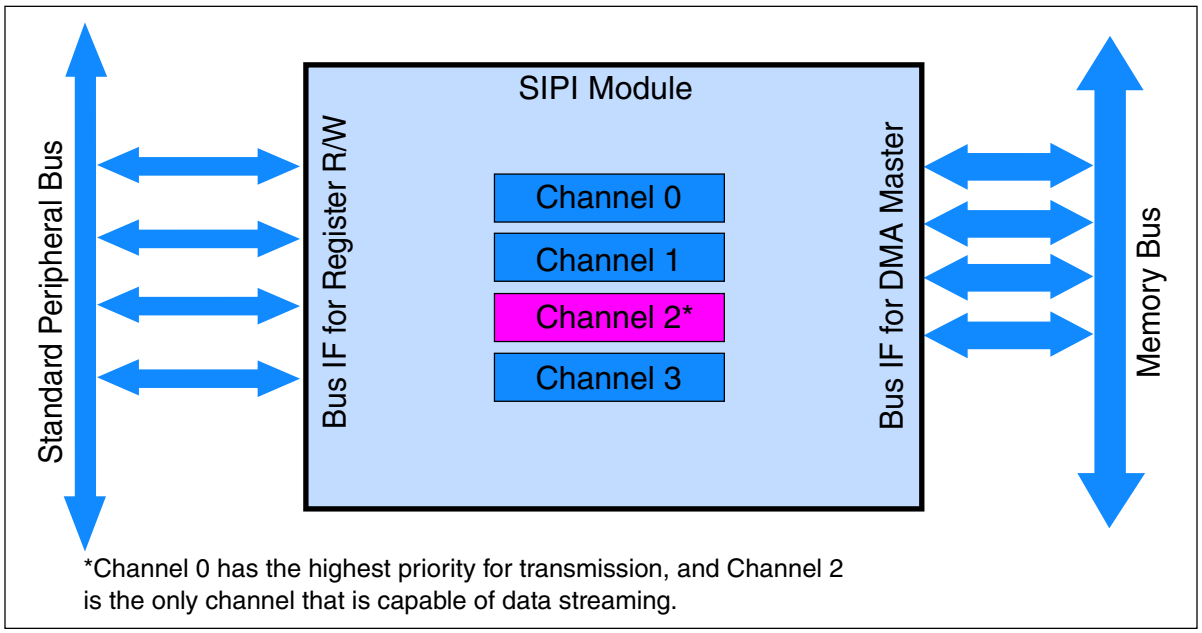
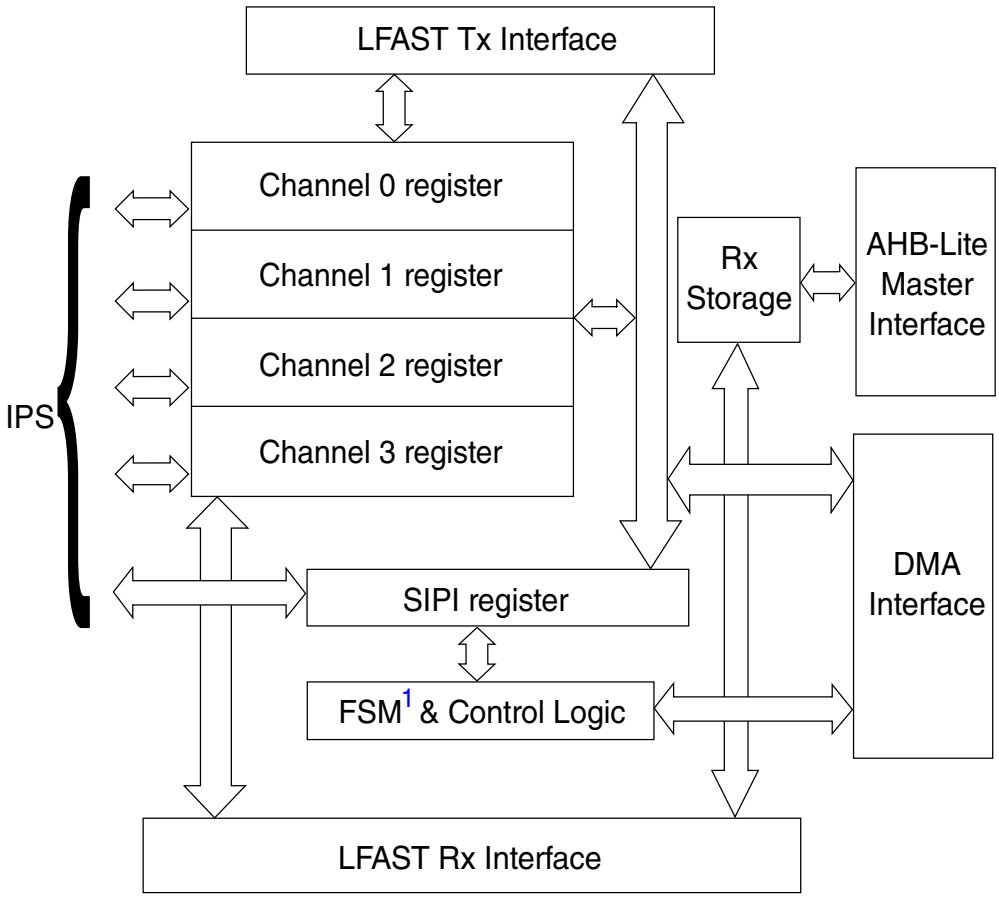


Figure 55-2. SIPI module

55.3 SIPI block diagram



[1] Finite State Machine

Figure 55-3. SIPI block diagram

55.4 Feature description

This section describes the features of the SIPI module.

55.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface which is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

55.4.2 Standard features

- IPS bus interface (PBRIDGE)
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

55.5 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

55.6 Functional description

55.6.1 External signals

The SIPI has no chip external signals.

55.6.2 Frame format

All frames have the same general format:

- 16 bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

55.6.2.1 Register read request

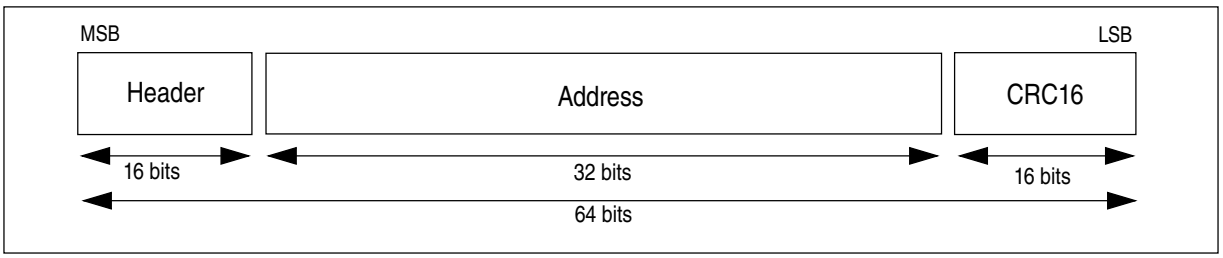


Figure 55-4. SIPI register read request

55.6.2.2 Register read response, ID request response

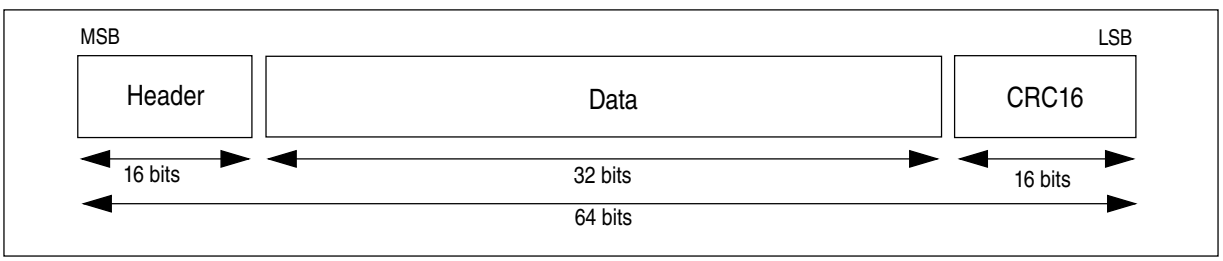


Figure 55-5. SIPI read response

55.6.2.3 Register write request

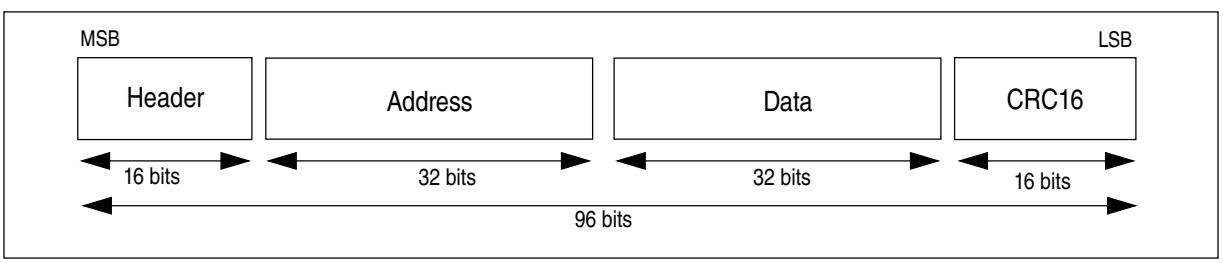


Figure 55-6. SIPI register write request

55.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

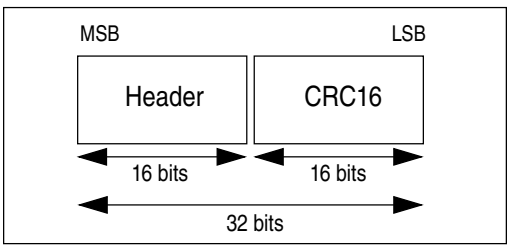


Figure 55-7. SIPI write acknowledge

55.6.2.5 Streaming write request

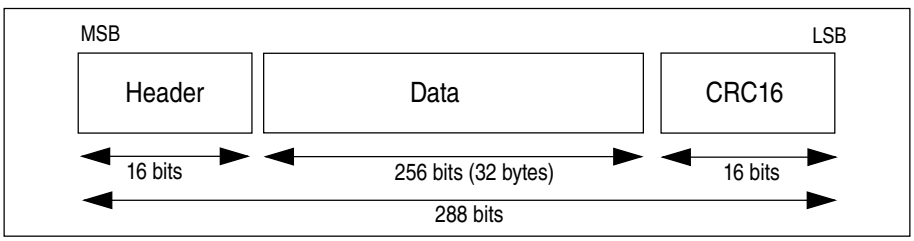


Figure 55-8. Streaming write request format

NOTE

Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in Figure 55-8.

55.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

55.6.2.6.1 SIPI header coding

Figure 55-9, Table 55-1, Figure 55-10, and Table 55-2 show how the SIPI header bits are coded.

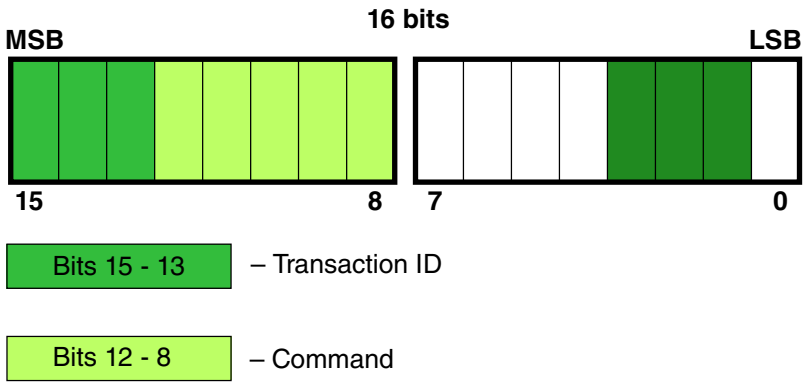


Figure 55-9. SIPI header coding

Table 55-1 shows the command coding for the bits[12:8] of the header.

Table 55-1. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
…	…	…
1F	Reserved	—

Figure 55-10 shows the channel number field in the header and Table 55-2 shows the channel number coding.

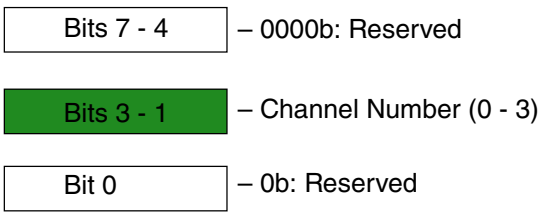


Figure 55-10. SIPI header channel field

Table 55-2. SIPI header channel number coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select field (SIPI_MCR[CHNSB])		
	Code table I (SIPI_MCR[CHNSB] = 1)	Code table II (SIPI_MCR[CHNSB] = 0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

55.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

55.6.2.8 Payload field

Table 55-3 shows the payload sizes of LFAST frames.

Table 55-3. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16

Table continues on the next page...

Table 55-3. Payload size of LFAST channel frame (continued)

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 55-4 shows the converted coding of LFAST for a given SIPI code.

Table 55-4. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex)[1]
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

55.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). MSB is sent first in the data stream.

55.6.2.9.1 CRC field examples

55.6.2.9.1.1 Example 1 – 32 bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

55.6.2.9.1.2 Example 2 – 32 bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

55.6.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

55.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

55.7.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

55.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in Figure 55-11.

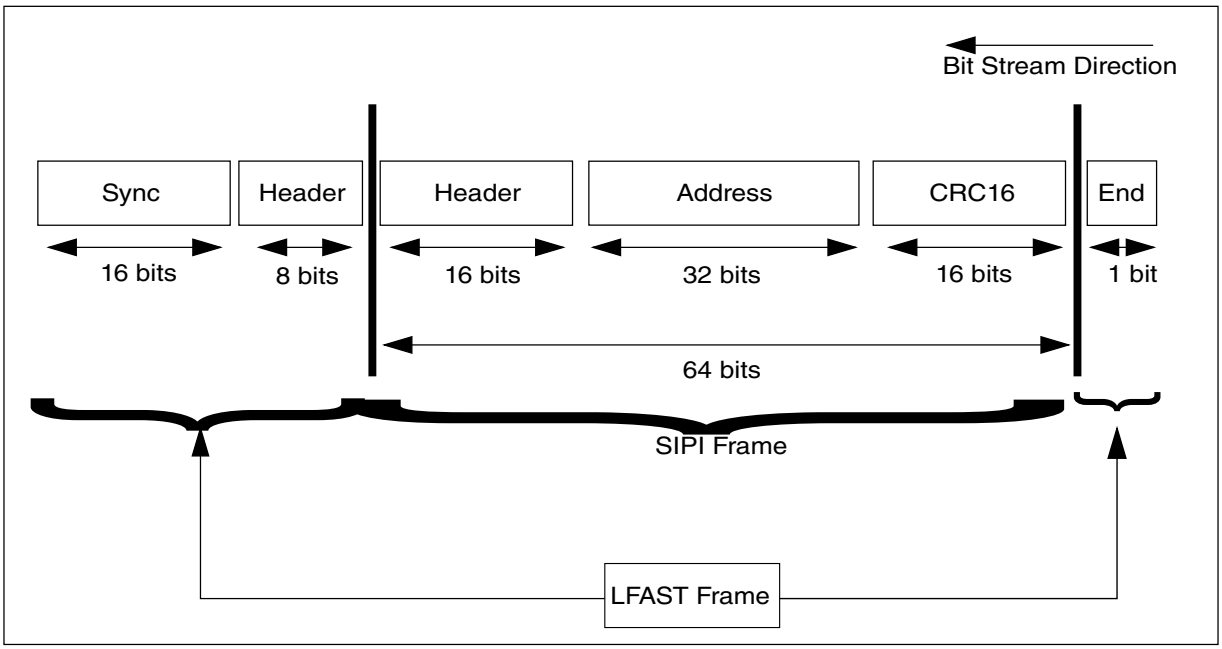


Figure 55-11. Read request transfer

55.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see Figure 55-12). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see Figure 55-13).

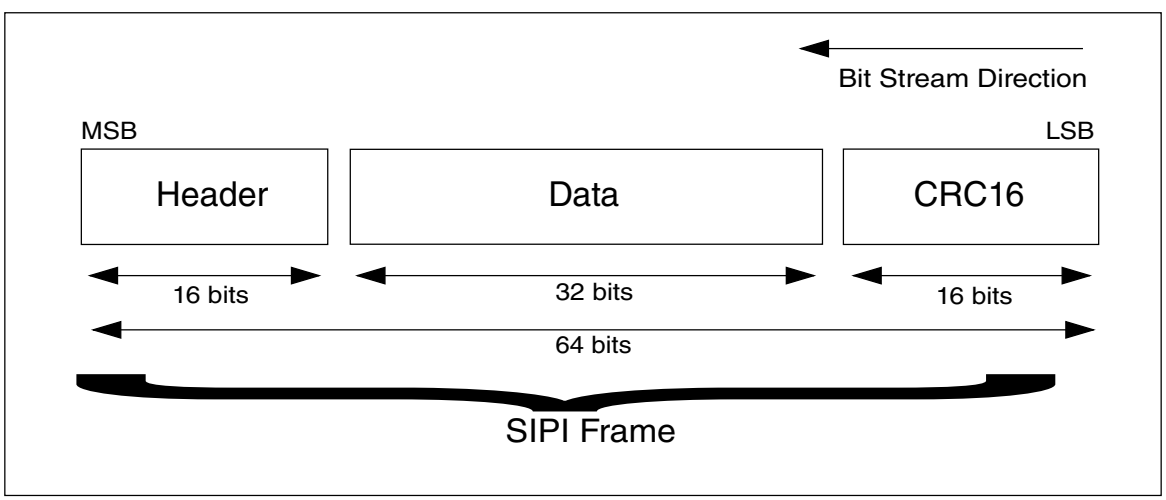


Figure 55-12. Read answer transfer

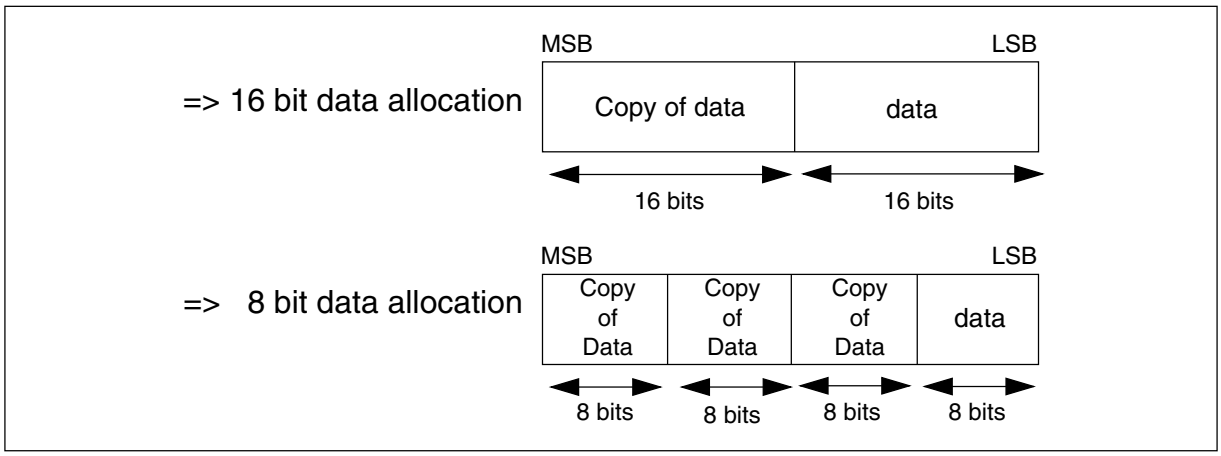


Figure 55-13. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
  - Bit[0]:
    - 0 - halfword 1 enabled (MSB)
    - 1 - halfword 0 enabled (LSB)

55.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see Normal write transfer) as shown in Figure 55-14.

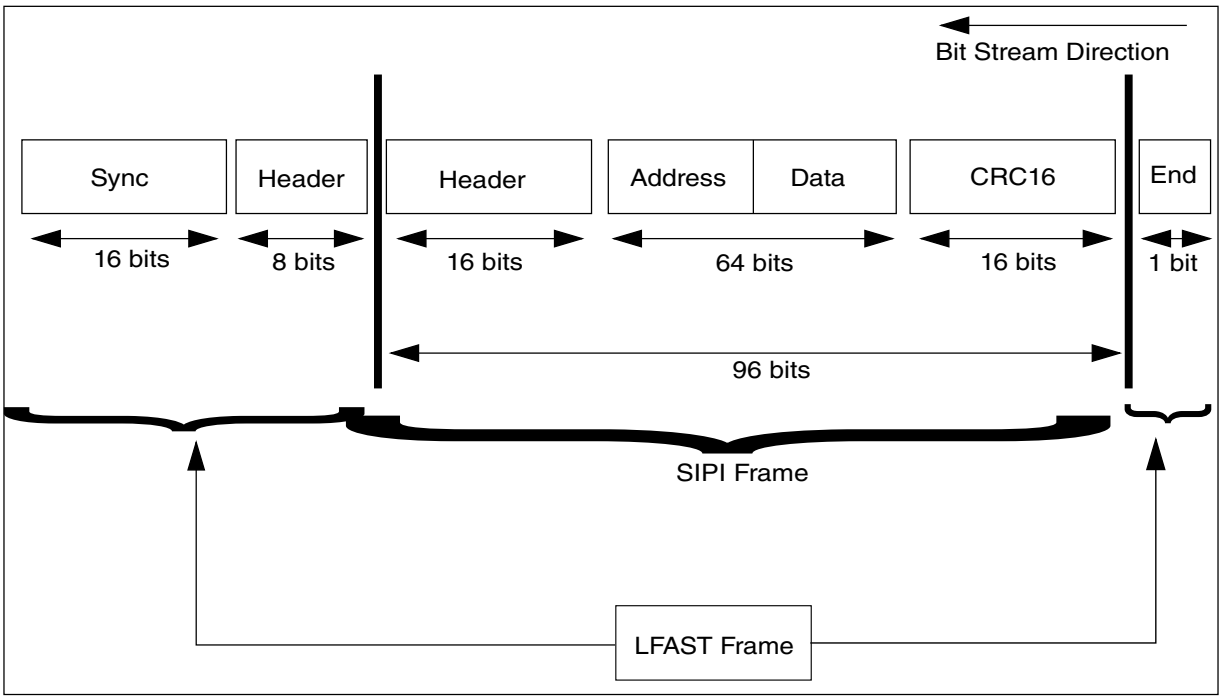


Figure 55-14. Register write transfer (showing LFAST frame encapsulation)

55.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in Figure 55-15.

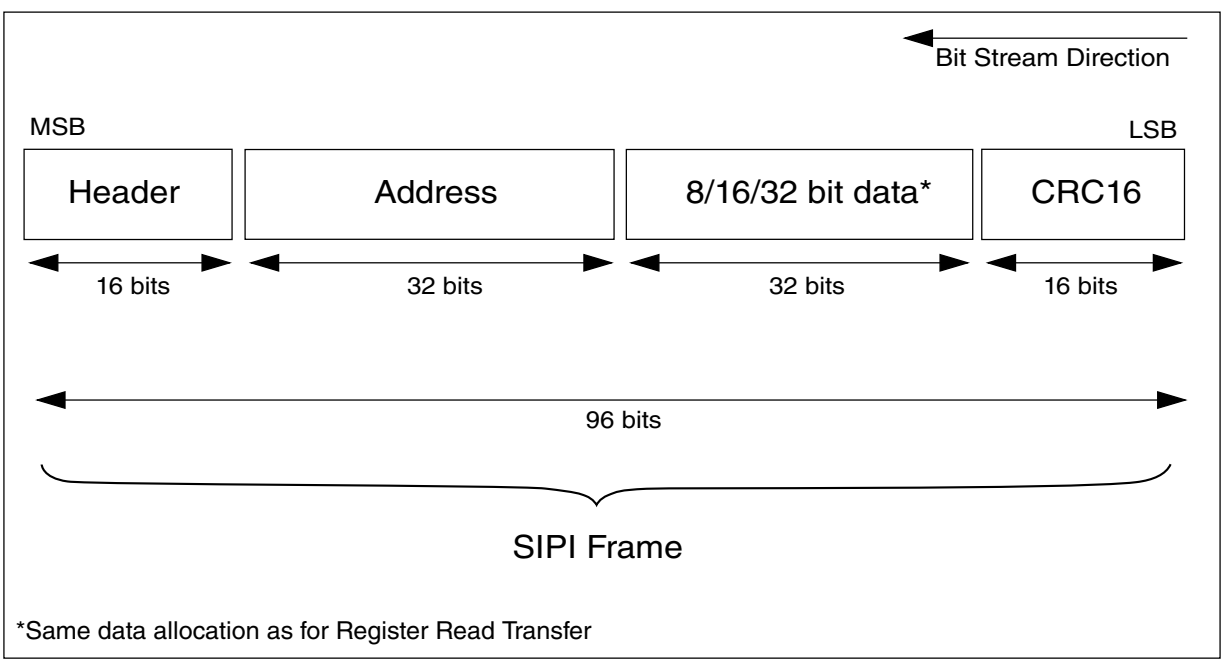


Figure 55-15. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
  - Bit[0]:
    - 0 - half-word 1 enabled (MSB)
    - 1 - half-word 0 enabled (LSB)

55.7.3.2 Streaming write transfer

Streaming write transfer has 256 bits of payload.

55.7.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. The SIPI Maximum Count Register (SIPI_MAXCR) and SIPI Address Reload Register (SIPI_ARR) are written before the SIPI Address Count Register (SIPI_ACR). This is to avoid unwanted behavior of the SIPI attempting to access non-shared memory. This is only true for the very first streaming command, subsequent streaming command(s) may not need to write the SIPI_MAXCR and SIPI_ARR (see SIPI Max Count Register (SIPI_MAXCR), SIPI Address Count Register (SIPI_ACR) and SIPI Address Reload Register (SIPI_ARR)).

55.7.3.2.2 Streaming transfer with data

Figure 55-16 shows the packet structure of the streaming transfer containing data.

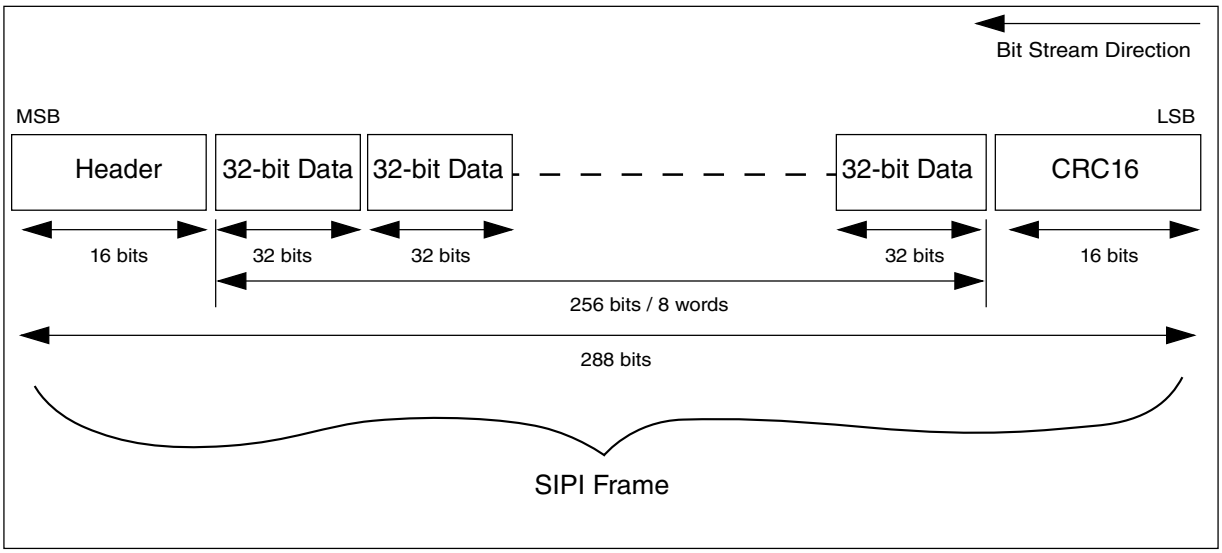


Figure 55-16. Streaming transfer with data

55.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see Write Aknowledge transfer. The CRC bits are calculated on the header field.

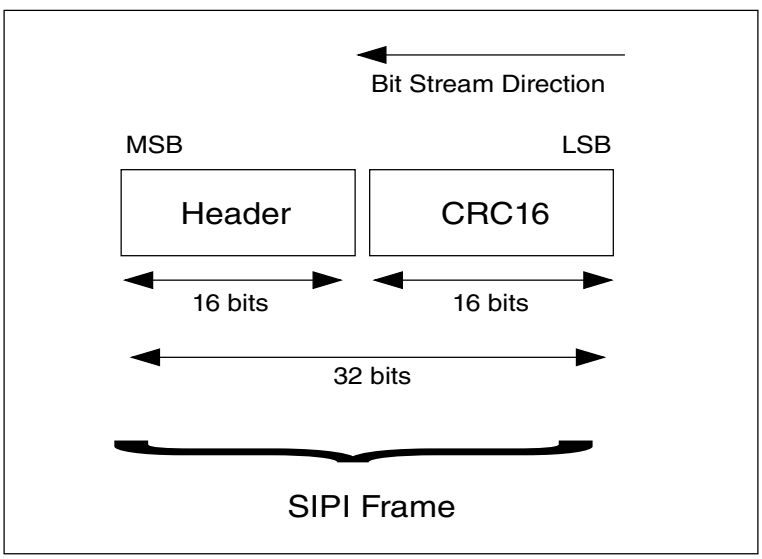


Figure 55-17. Write Aknowledge transfer (LFAST encapsulation is not shown)

55.7.5 ID request response

55.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in Figure 55-18.

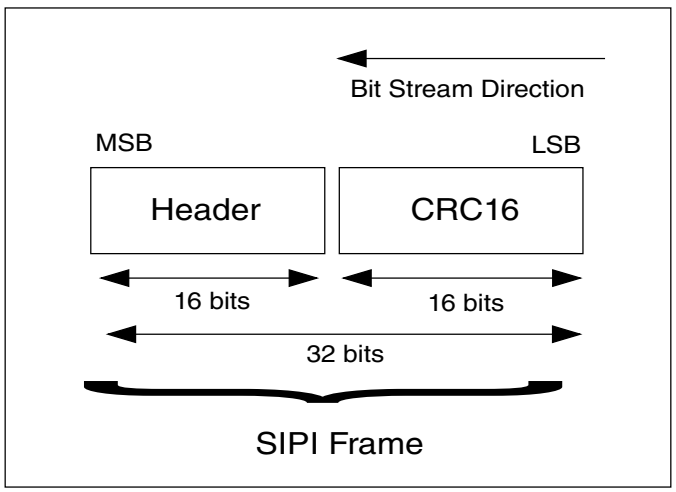


Figure 55-18. ID request transfer (LFAST encapsulation is not shown)

55.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in Figure 55-19.

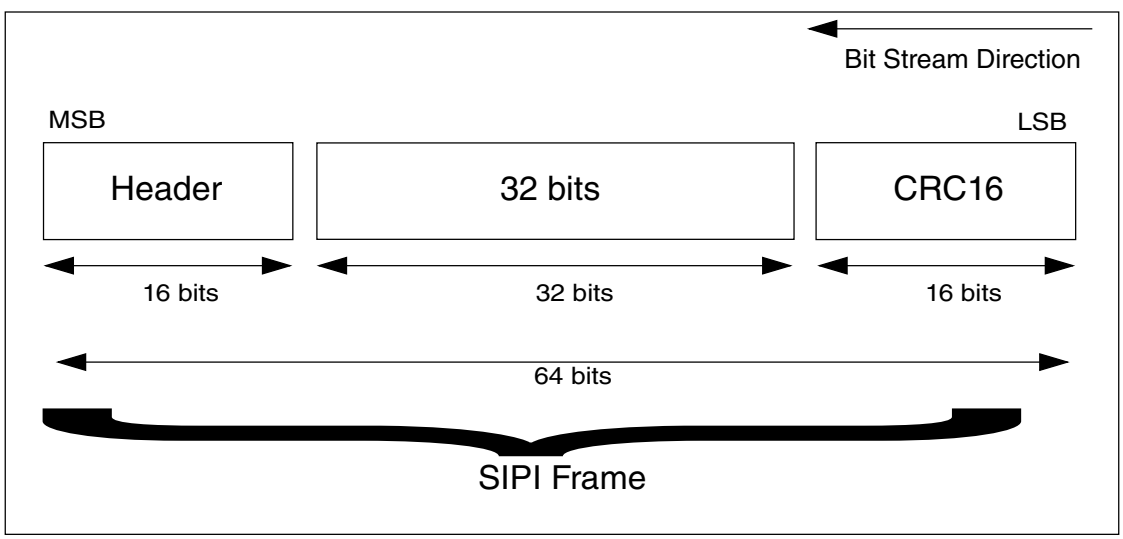


Figure 55-19. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

55.8 Transfer API and flow charts

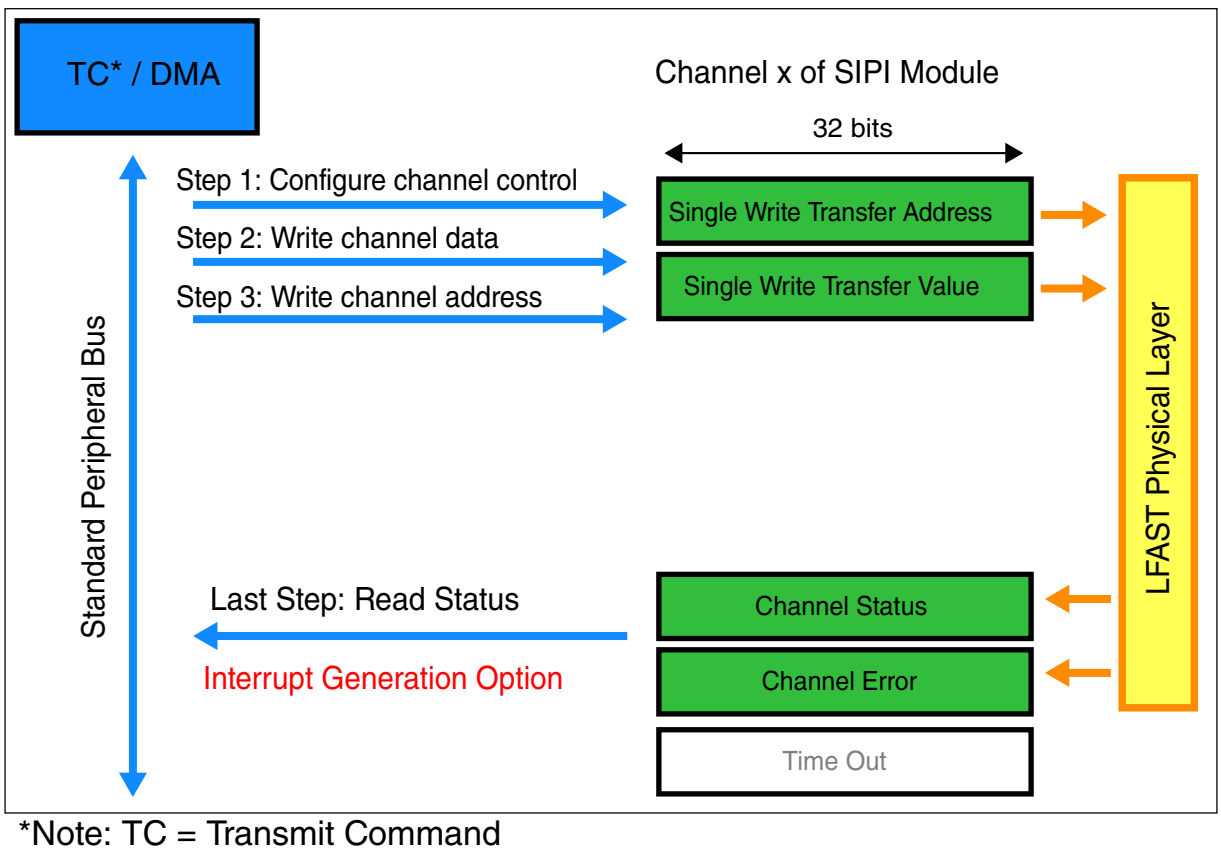


Figure 55-20. SIPI single register write API

Implement the following steps to generate a single Write Transfer Request (Figure 55-20):

1. Configure data and SIPI_CCR*n* at the initiator node.
2. Configure SIPI_CAR*n* at the initiator node.

   As soon as the channel address register (SIPI_CAR*n*) is written and if CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
3. Software polls the status register bits (SIPI_CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single Write transfer request reception (Figure 55-21):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

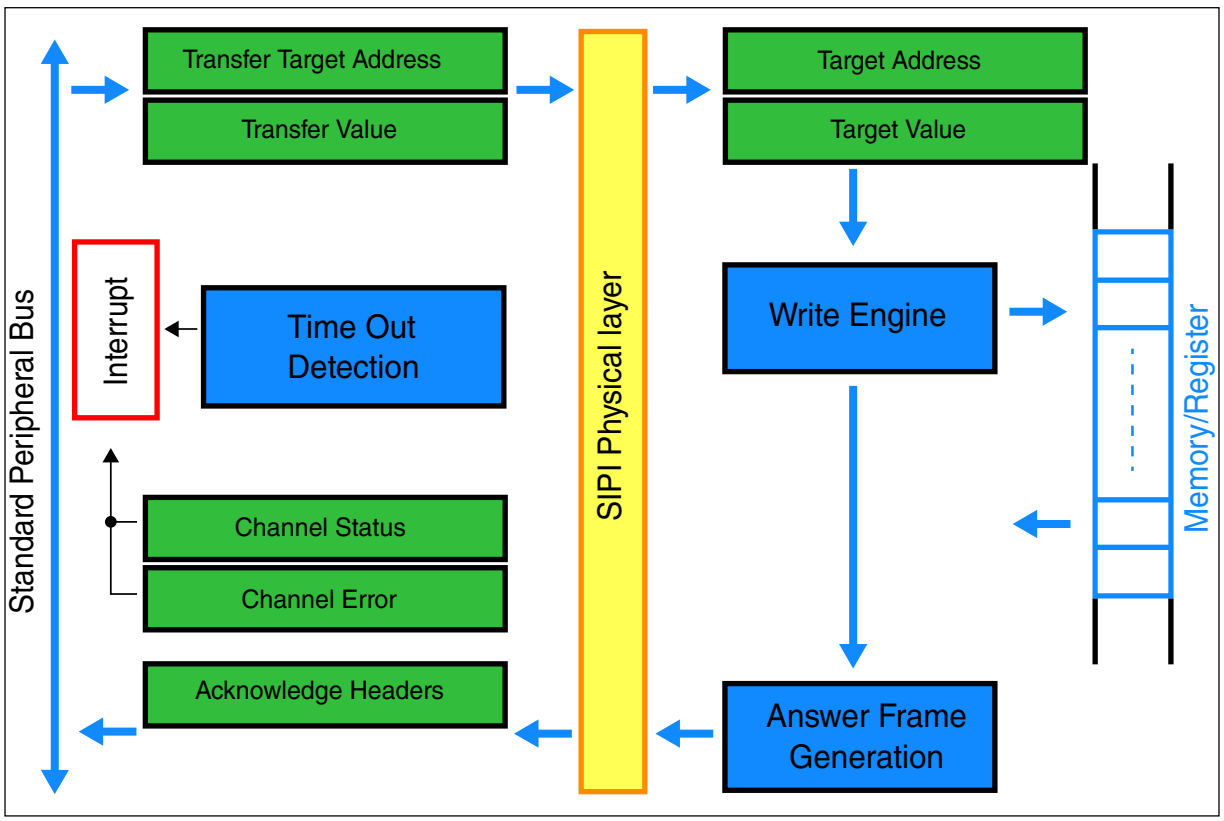


Figure 55-21. SIPI Single Register Write API – Flow Chart

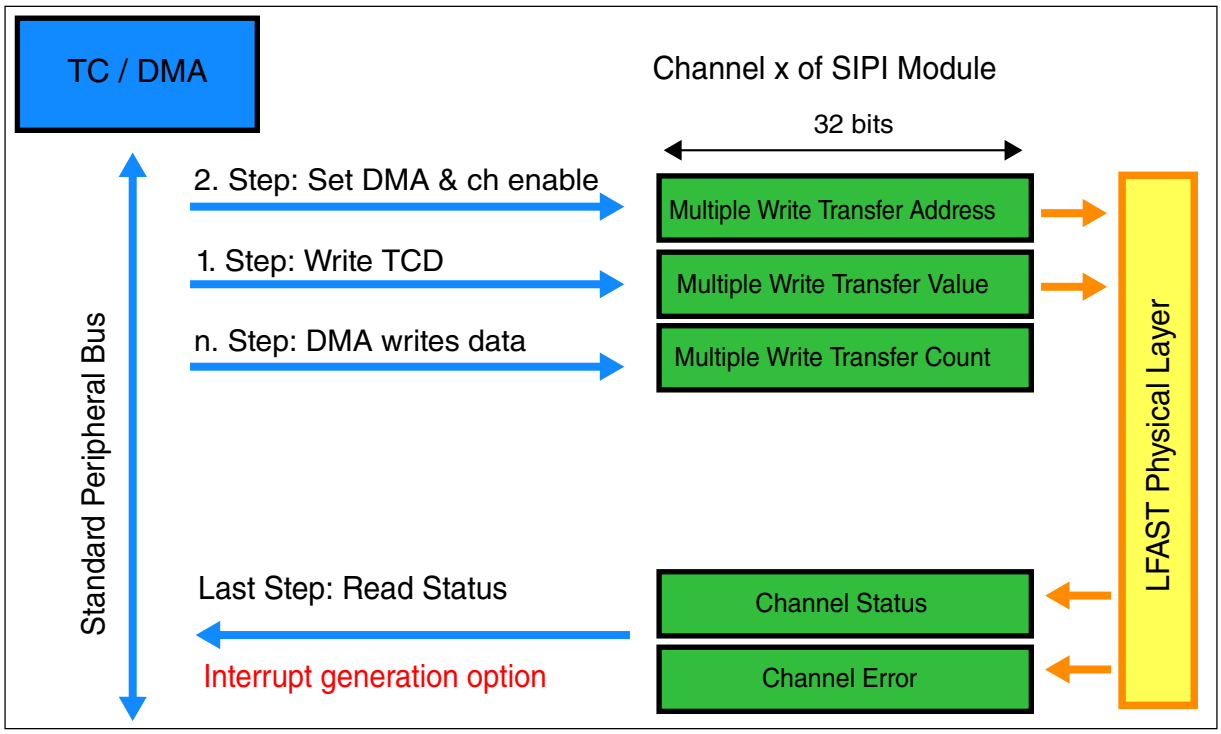


Figure 55-22. SIPI Multiple Register Write API

For multiple write transfer request generation (Figure 55-22):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DAN] = 1.
3. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size. SIPI_CDR2_0 should be written with the MSB.
4. When the copying process is completer, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
5. Software should poll the SIPI_CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
6. If SIPI_CCR*n*[DEN] = 1, the SIPI request will transfer to the DMA controller. If not, the state machine go idle.
7. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (Figure 55-22):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, target node will generate an acknowledge frame and send it back to the LFAST.

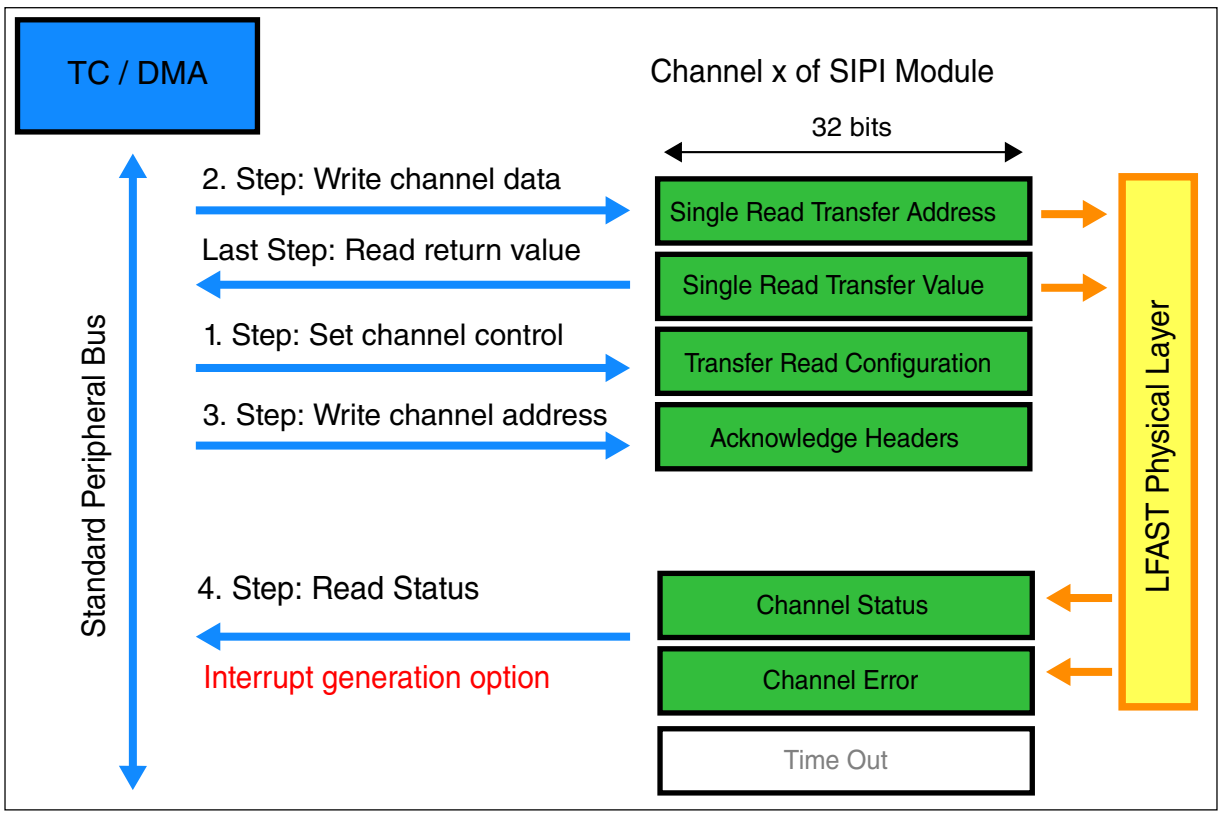


Figure 55-23. SIPI Single Register Read API

For Single Read Transfer Request generation (Figure 55-23):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, an interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1 which indicates a timeout has occured.
4. If SIPI_CSR*n*[RAR] = 1 then software can read the data register, or can other necessary action if SIPI_CSR*n*[RAR] = 0.

On Single Read transfer request reception (Figure 55-23):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

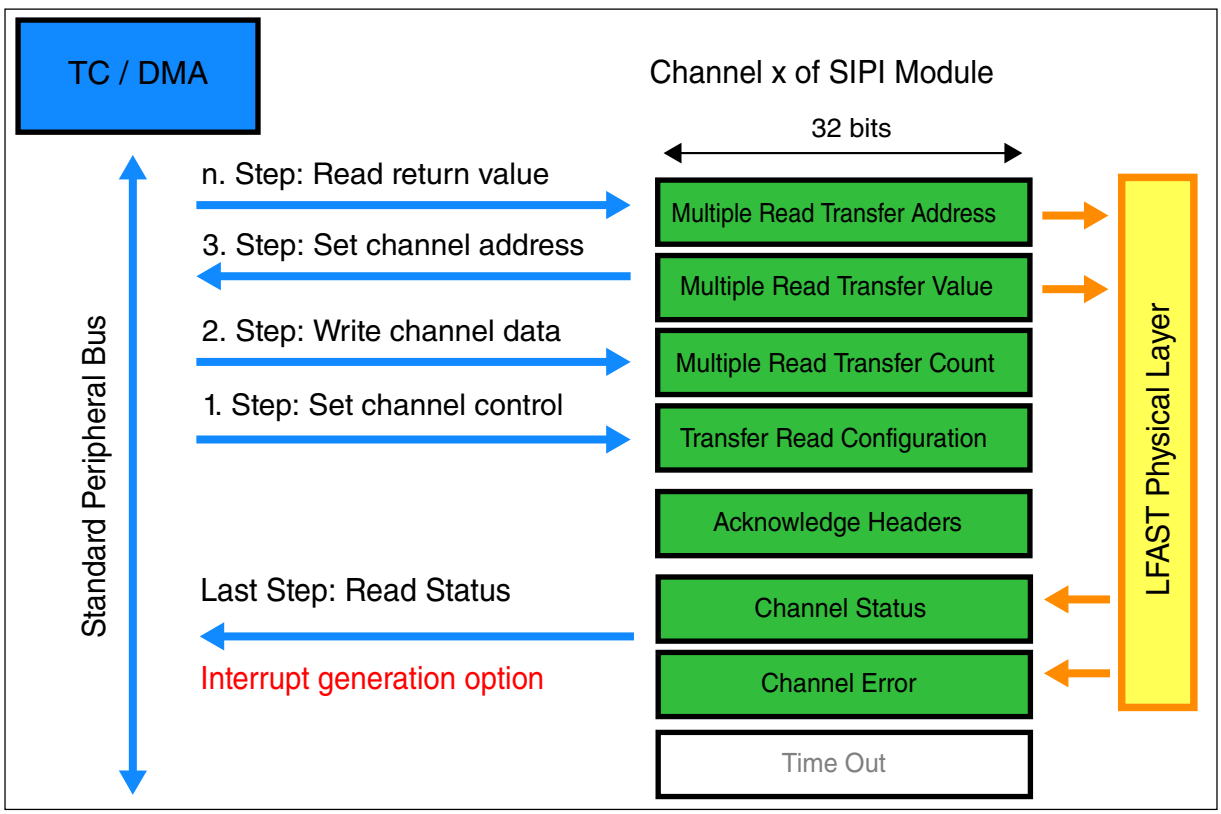


Figure 55-24. SIPI multiple register read API

For multiple read transfer request generation (Figure 55-24):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then starts transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1 an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1 which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (Figure 55-24):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

NOTE

Read answer interrupt pins are also cleared by hardware in DMA transfers. This can lead the software in trouble where it tracks the requests by keep counting the number of times the interrupt was received and cleared.

Figure 55-25, Figure 55-26, Figure 55-27 and Figure 55-28 show the SIPI data streaming of data.

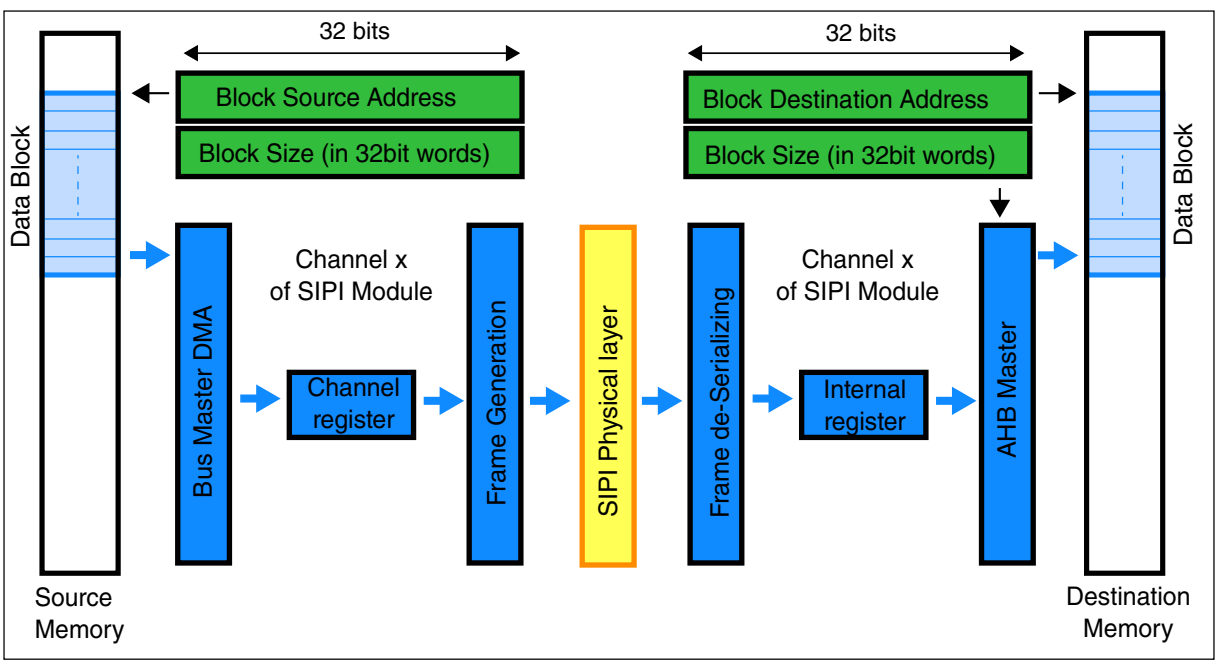


Figure 55-25. SIPI data stream model

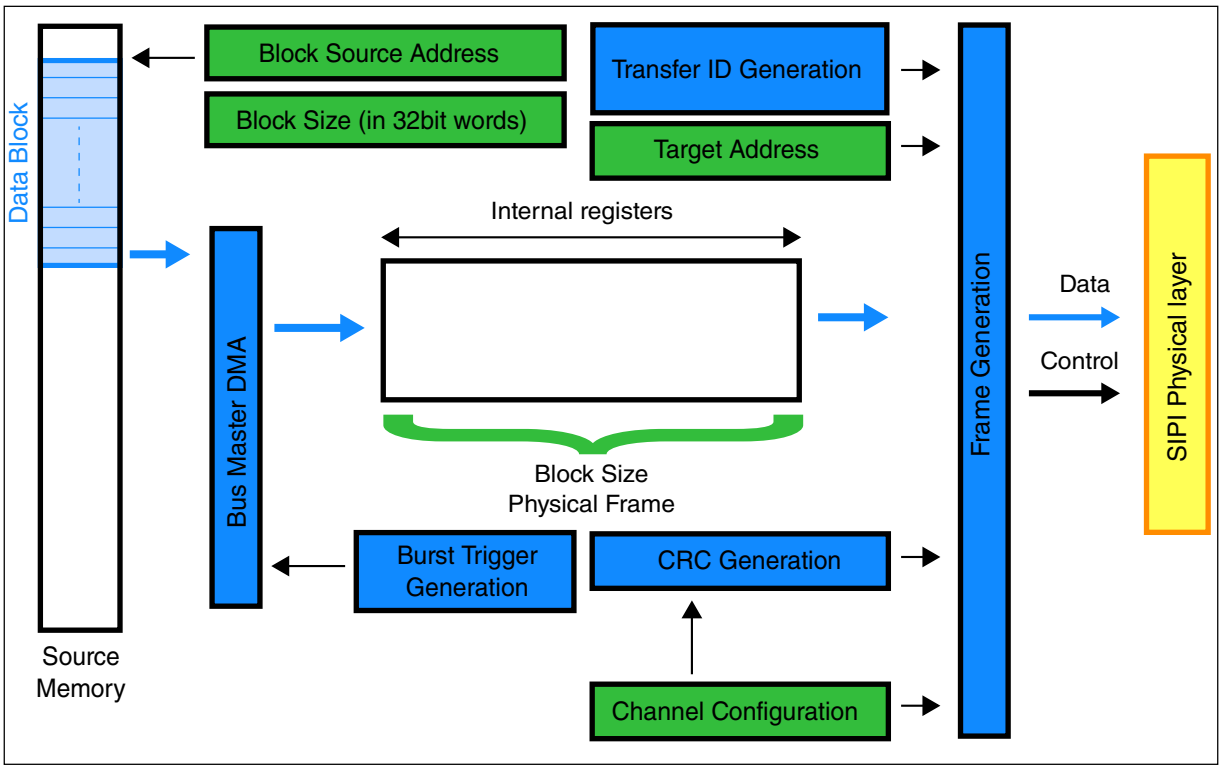


Figure 55-26. SIPI data stream model - Initiator

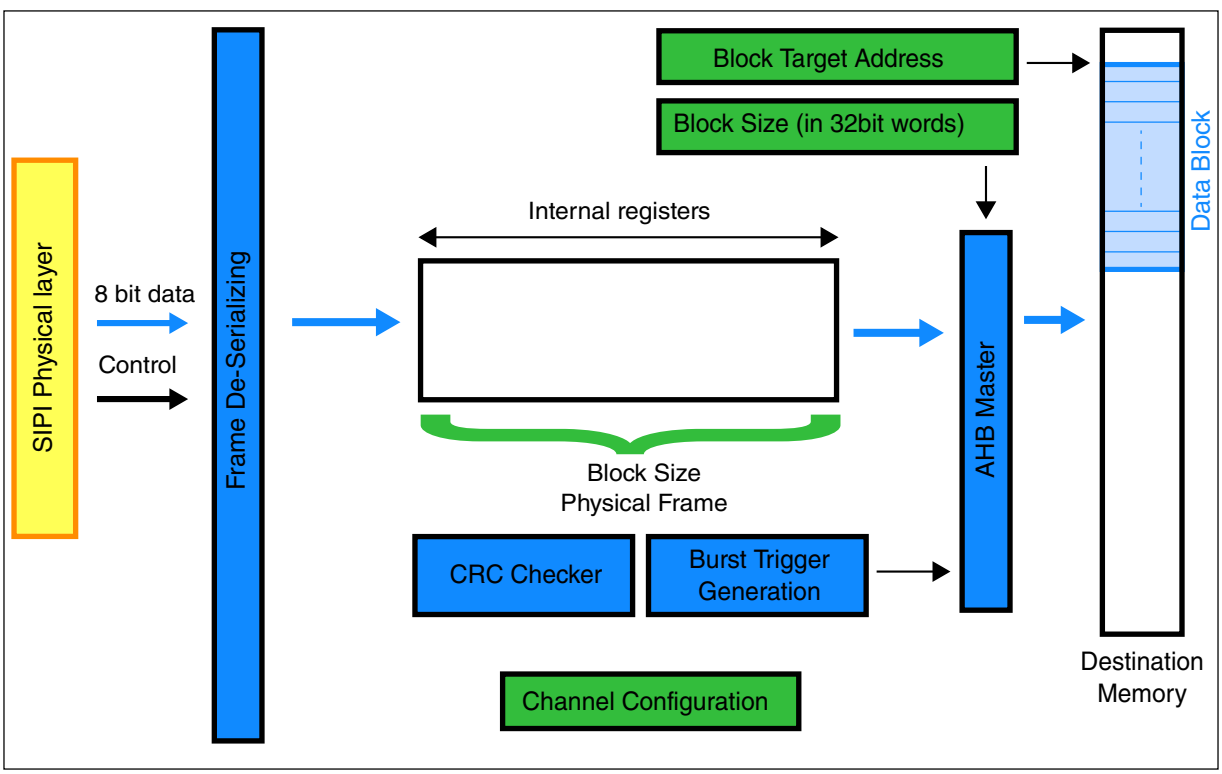


Figure 55-27. SIPI data stream model - Target

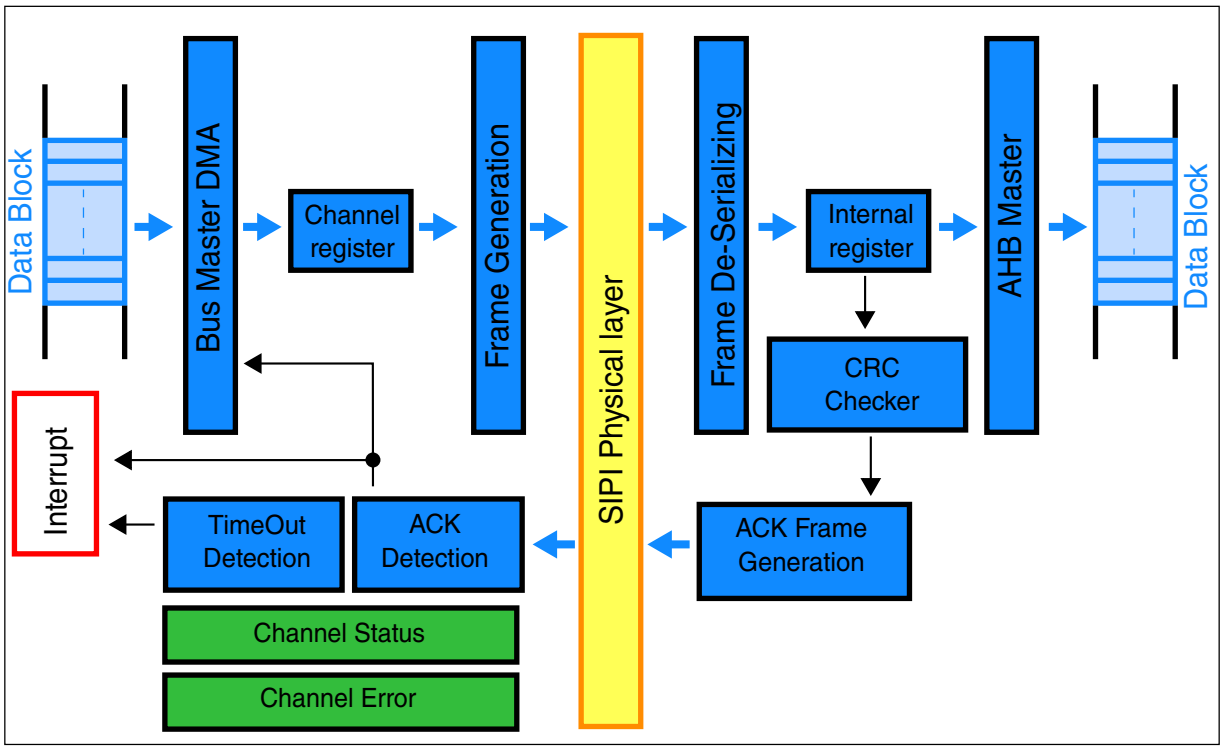


Figure 55-28. SIPI data stream with acknowledge trigger

55.9 DMA programming sequence

The DMA programming sequence is as follows:

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. When SIPI receives read response and LFAST's error flags are not asserted, it initiates a DMA request. In case the LFAST asserts error flag, SIPI moves to step 8.
6. When SIPI initiates DMA request, the DMA transfers data between SIPI and RAM.
7. If DMA transfer minor loop/major loop is complete, and the request is negated, ipd_request is negated.
8. If an error occurs:

- The SIPI generates an interrupt
- Software takes control of the SIPI

9. Steps 5 – 8 are repeated until the transfer is finished.

55.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

55.10.1 Initialization mode

To enter Initialization mode software writes SIPI_MCR[INIT]=1 SIPI_MCR[MOEN] must be set before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT]=0 (see SIPI Module Configuration Register (SIPI_MCR)).

Note

It is recommended that software checks the state of SIPI (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

55.10.2 Normal mode

Once software has completed initialization of SIPI (SIPI_MCR[INIT]=1), it can enter Normal mode by writing SIPI_MCR[INIT]=0. SIPI needs to be in Normal mode for all data transfers (see SIPI Module Configuration Register (SIPI_MCR)).

Note

SIPI must be enabled before a attempting to write SIPI_MCR[INIT] (SIPI_MCR[MOEN]=1).

55.10.3 Module Disable (MD)

MD mode in the SIPI is used to help reduce power consumption. By default, SIPI is in Disable mode, SIPI_MCR[MOEN]=0, and is exited by writing SIPI_MCR[MOEN]=1 (see SIPI Module Configuration Register (SIPI_MCR)). Future SIPI Tx transfers are disabled in Module Disable mode. To disable Rx functionality, the Target Enable (TEN) bit field has to be deasserted, then the internal Rx state machine resets to initial state.

55.11 Errors

This section describes the potential errors that can occur during SIPI operation.

55.11.1 Timeout error

A timeout error is generated at the initiator node when the acknowledge/response is not received within the time configured in the corresponding CTOR*n*[TOR] field setting (see SIPI Channel Timeout Register 0 (SIPI_CTOR0), SIPI Channel Timeout Register 1 (SIPI_CTOR1), SIPI Channel Timeout Register 2 (SIPI_CTOR2) and SIPI Channel Timeout Register 3 (SIPI_CTOR3)). A timeout error is indicated when ERR[TOE*n*]=1 (see SIPI Error Register (SIPI_ERR)).

Note

SIPI should not drop the response even after a timeout occurs. Software will poll both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may then be discarded.

55.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match the calculated CRC, and the SR[GCRCE] is set (see SIPI Status Register (SIPI_SR)). An interrupt will be asserted if MCR[CRCIE]=1 (see SIPI Module Configuration Register (SIPI_MCR)).

Note

The target node will not send an acknowledge to the initiator node when a CRC error is generated. An interrupt will be generated on the target side if the corresponding interrupt

enable bit is set. The initiator node will detect a timeout and take necessary action.

55.11.3 Maximum count reached error

The maximum count reached error is only generated at the target node. It is generated when the value of the SIPI_ACR is equal to the SIPI_MAXCR (see SIPI Max Count Register (SIPI_MAXCR) and SIPI Address Count Register (SIPI_ACR)). When the maximum count is reached, SIPI_SR[MCR] = 1 (see SIPI Status Register (SIPI_SR)). An interrupt will be generated if SIPI_MCR[MCRIE] = 1 (see SIPI Module Configuration Register (SIPI_MCR)).

55.11.4 Transaction ID error

The Transaction ID (TID) error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR*n*[TID] (transaction ID bits, see SIPI Channel Status Register 0 (SIPI_CSR0), SIPI Channel Status Register 1 (SIPI_CSR1), SIPI Channel Status Register 2 (SIPI_CSR2), and SIPI Channel Status Register 3 (SIPI_CSR3)). SIPI_CSR*n*[TIDE] = 1 when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

55.11.5 Acknowledge error

An incorrect acknowledge is received only from the initiator. When the acknowledge received is incorrect, SIPI_ERR[ACKR*n*] will be set (see SIPI Error Register (SIPI_ERR)). An interrupt will be generated if SIPI_CIR*n*[WAIE]=1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

55.12 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

5) Data will be sent as FF77_BB33h.

55.13 Interrupt logic

A description of the interrupt logic can be found in the following figure.

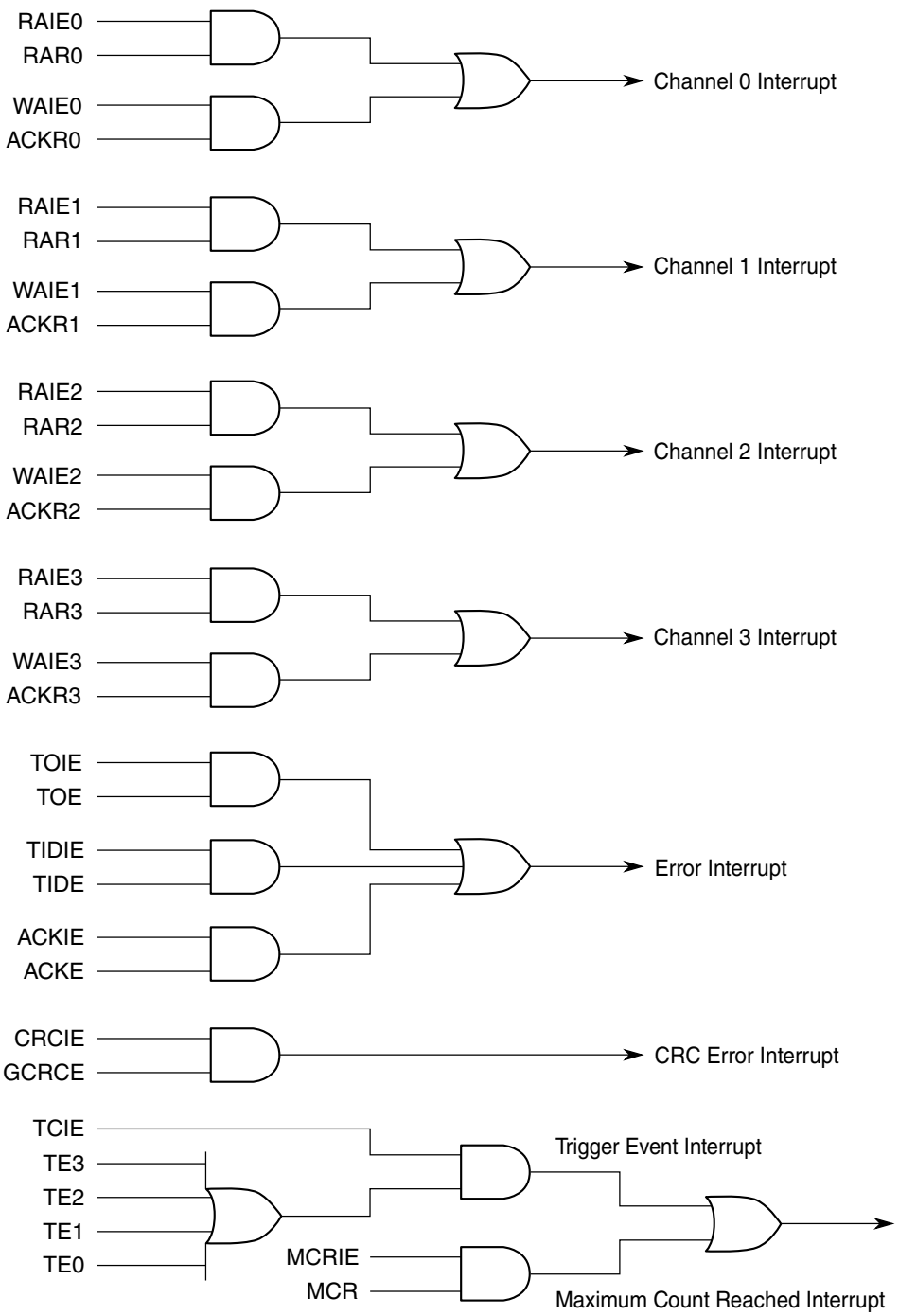


Figure 55-29. Interrupt description

55.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

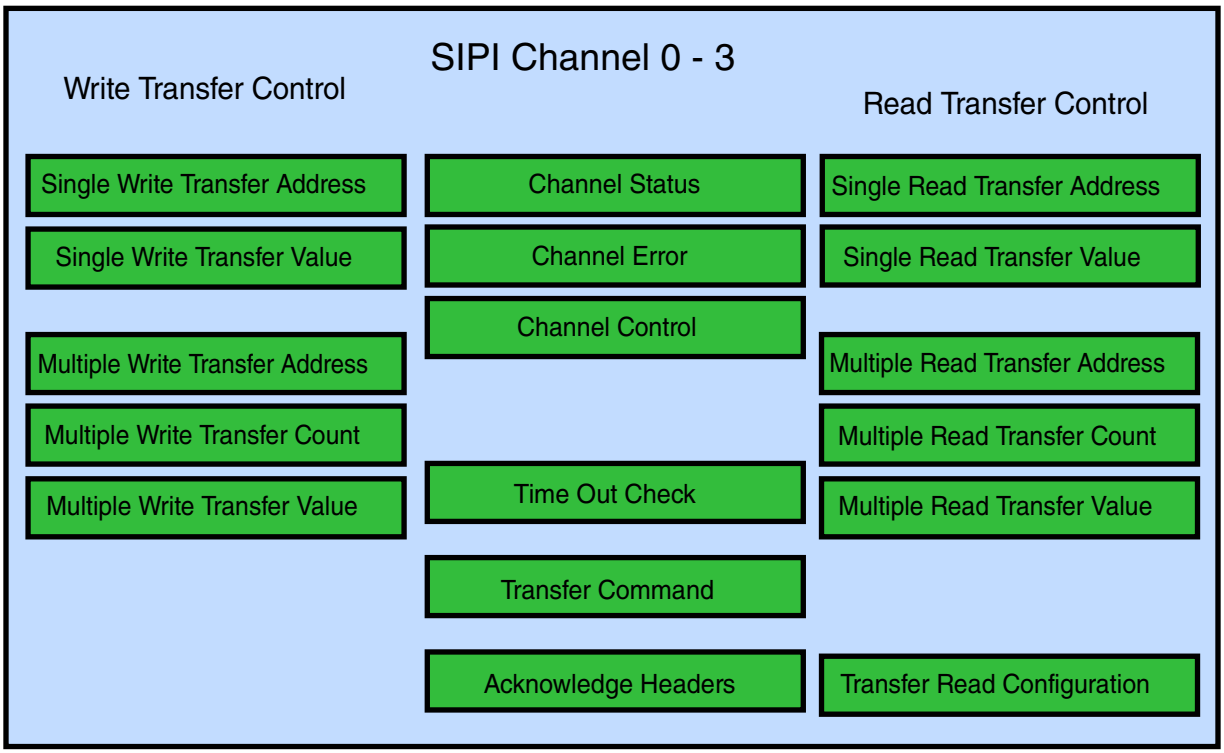


Figure 55-30. SIPI control and status overview – Register transfer

55.15 Memory map and register definition

SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIPI Channel Control Register 0 (SIPI_CCR0)	32	R/W	0000_0000h	55.15.1/ 2543
4	SIPI Channel Status Register 0 (SIPI_CSR0)	32	R	0000_0000h	55.15.2/ 2546
C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	32	R/W	0000_0000h	55.15.3/ 2547
10	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	32	R/W	0000_00FFh	55.15.4/ 2548
14	SIPI Channel CRC Register 0 (SIPI_CCRC0)	32	R	0000_0000h	55.15.5/ 2549
18	SIPI Channel Address Register 0 (SIPI_CAR0)	32	R/W	0000_0000h	55.15.6/ 2549
1C	SIPI Channel Data Register 0 (SIPI_CDR0)	32	R/W	0000_0000h	55.15.7/ 2550
20	SIPI Channel Control Register 1 (SIPI_CCR1)	32	R/W	0000_0000h	55.15.8/ 2550

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	SIPI Channel Status Register 1 (SIPI_CSR1)	32	R	0000_0000h	55.15.9/ 2553
2C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	32	R/W	0000_0000h	55.15.10/ 2555
30	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	32	R/W	0000_00FFh	55.15.11/ 2556
34	SIPI Channel CRC Register 1 (SIPI_CCRC1)	32	R	0000_0000h	55.15.12/ 2557
38	SIPI Channel Address Register 1 (SIPI_CAR1)	32	R/W	0000_0000h	55.15.13/ 2557
3C	SIPI Channel Data Register 1 (SIPI_CDR1)	32	R/W	0000_0000h	55.15.14/ 2558
40	SIPI Channel Control Register 2 (SIPI_CCR2)	32	R/W	0000_0000h	55.15.15/ 2558
44	SIPI Channel Status Register 2 (SIPI_CSR2)	32	R	0000_0000h	55.15.16/ 2561
4C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	32	R/W	0000_0000h	55.15.17/ 2563
50	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	32	R/W	0000_00FFh	55.15.18/ 2564
54	SIPI Channel CRC Register 2 (SIPI_CCRC2)	32	R	0000_0000h	55.15.19/ 2565
58	SIPI Channel Address Register 2 (SIPI_CAR2)	32	R/W	0000_0000h	55.15.20/ 2565
5C	SIPI Channel Data Register 2 (SIPI_CDR2_0)	32	R/W	0000_0000h	55.15.21/ 2566
60	SIPI Channel Data Register 2 (SIPI_CDR2_1)	32	R/W	0000_0000h	55.15.21/ 2566
64	SIPI Channel Data Register 2 (SIPI_CDR2_2)	32	R/W	0000_0000h	55.15.21/ 2566
68	SIPI Channel Data Register 2 (SIPI_CDR2_3)	32	R/W	0000_0000h	55.15.21/ 2566
6C	SIPI Channel Data Register 2 (SIPI_CDR2_4)	32	R/W	0000_0000h	55.15.21/ 2566
70	SIPI Channel Data Register 2 (SIPI_CDR2_5)	32	R/W	0000_0000h	55.15.21/ 2566
74	SIPI Channel Data Register 2 (SIPI_CDR2_6)	32	R/W	0000_0000h	55.15.21/ 2566
78	SIPI Channel Data Register 2 (SIPI_CDR2_7)	32	R/W	0000_0000h	55.15.21/ 2566
7C	SIPI Channel Control Register 3 (SIPI_CCR3)	32	R/W	0000_0000h	55.15.22/ 2566
80	SIPI Channel Status Register 3 (SIPI_CSR3)	32	R	0000_0000h	55.15.23/ 2569

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	32	R/W	0000_0000h	55.15.24/ 2571
8C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	32	R/W	0000_00FFh	55.15.25/ 2572
90	SIPI Channel CRC Register 3 (SIPI_CCRC3)	32	R	0000_0000h	55.15.26/ 2573
94	SIPI Channel Address Register 3 (SIPI_CAR3)	32	R/W	0000_0000h	55.15.27/ 2573
98	SIPI Channel Data Register 3 (SIPI_CDR3)	32	R/W	0000_0000h	55.15.28/ 2574
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	55.15.29/ 2574
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	55.15.30/ 2577
A4	SIPI Max Count Register (SIPI_MAXCR)	32	R/W	FFFF_FFFCh	55.15.31/ 2579
A8	SIPI Address Reload Register (SIPI_ARR)	32	R/W	0000_0000h	55.15.32/ 2579
AC	SIPI Address Count Register (SIPI_ACR)	32	R/W	0000_0000h	55.15.33/ 2580
B0	SIPI Error Register (SIPI_ERR)	32	R	0000_0000h	55.15.34/ 2581

55.15.1 SIPI Channel Control Register 0 (SIPI_CCR0)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR0[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR0[CB] = 0).

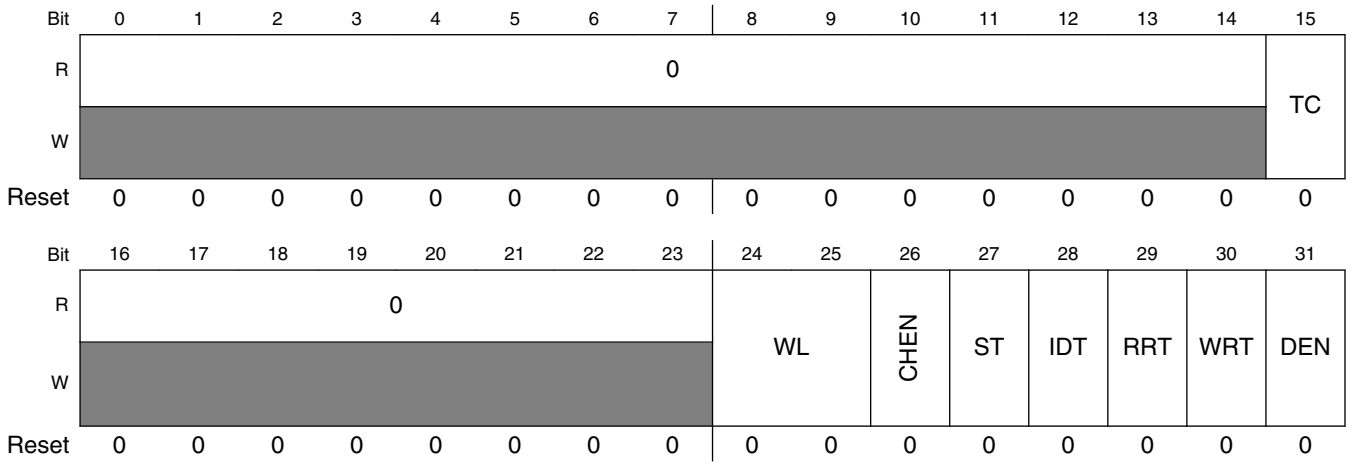
PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 0 (SIPI_CSR0)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 0 (SIPI_CAR0)).

Address: 0h base + 0h offset = 0h



SIPI_CCR0 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR0. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming.

Table continues on the next page...

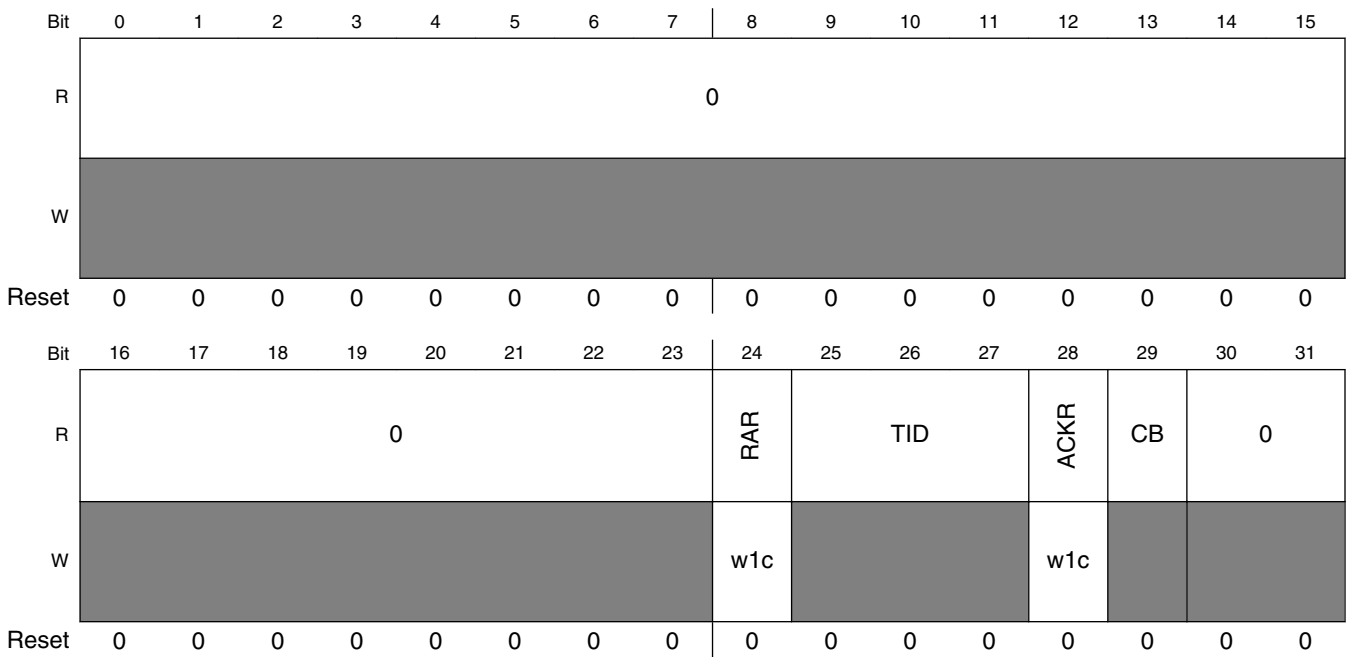
SIPI_CCR0 field descriptions (continued)

Field	Description
	0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1 or SIPI_CCR0[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

55.15.2 SIPI Channel Status Register 0 (SIPI_CSR0)

CSR0 contains the status bits for the current transfer.

Address: 0h base + 4h offset = 4h



SIPI_CSR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 0 status. 0 Channel 0 free 1 Channel 0 busy

Table continues on the next page...

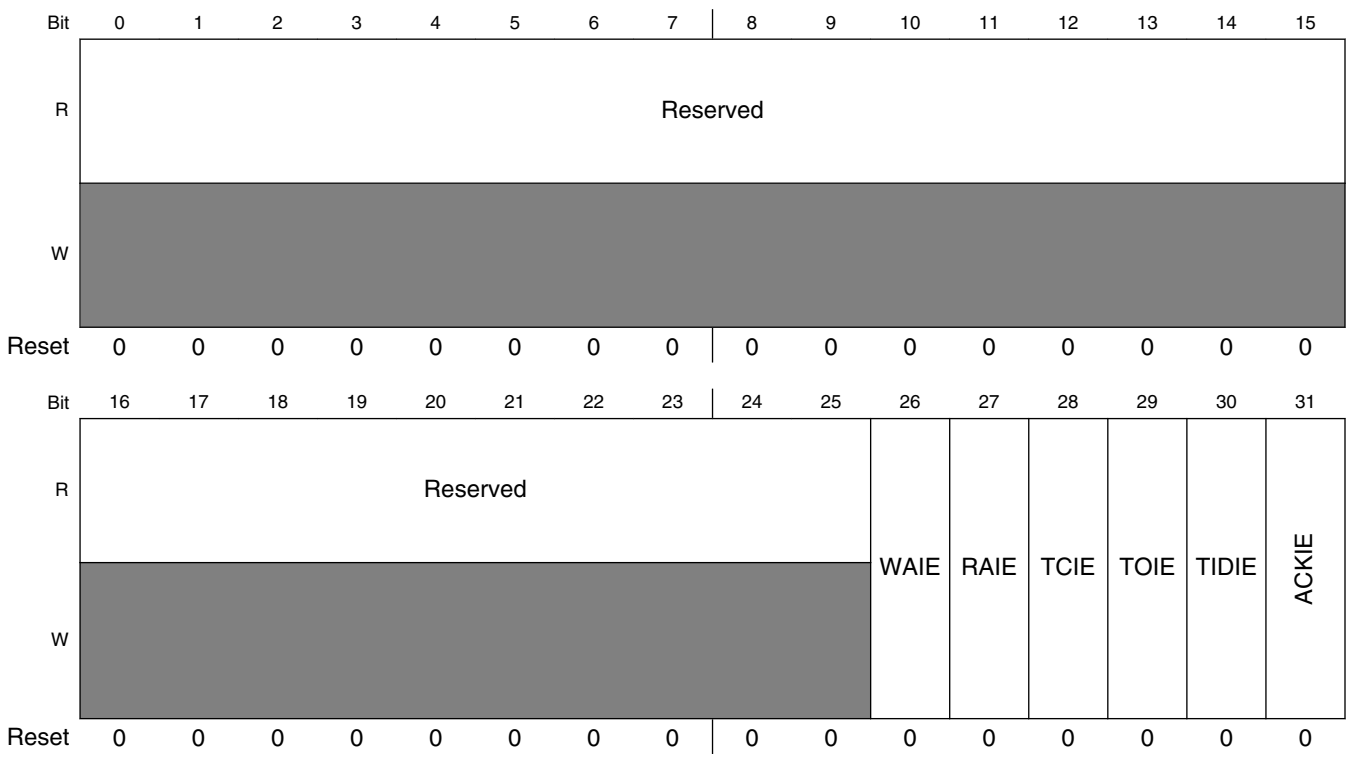
SIPI_CSR0 field descriptions (continued)

Field	Description
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.15.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)

SIPI_CIR0 contains the interrupt enable bits for channel 0.

Address: 0h base + Ch offset = Ch



SIPI_CIR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

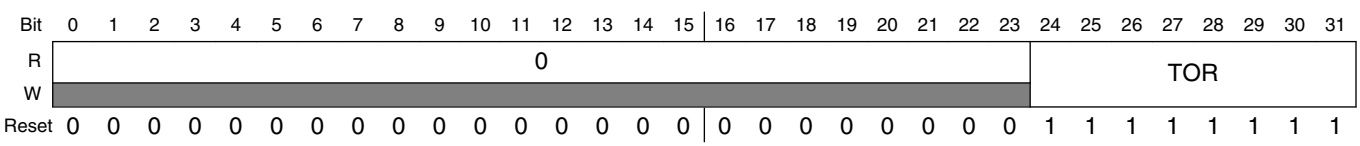
SIPI_CIR0 field descriptions (continued)

Field	Description
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

55.15.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)

SIPI_CTOR0 contains the timeout value for Tx requests.

Address: 0h base + 10h offset = 10h



SIPI_CTOR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

55.15.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC0 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

55.15.6 SIPI Channel Address Register 0 (SIPI_CAR0)

SIPI_CAR0 is the address target for data transmission.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR0 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

55.15.7 SIPI Channel Data Register 0 (SIPI_CDR0)

SIPI_CDR0 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR0 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

55.15.8 SIPI Channel Control Register 1 (SIPI_CCR1)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR1[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR1[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 1 (SIPI_CSR1)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 1 (SIPI_CAR1)).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR1 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR1. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

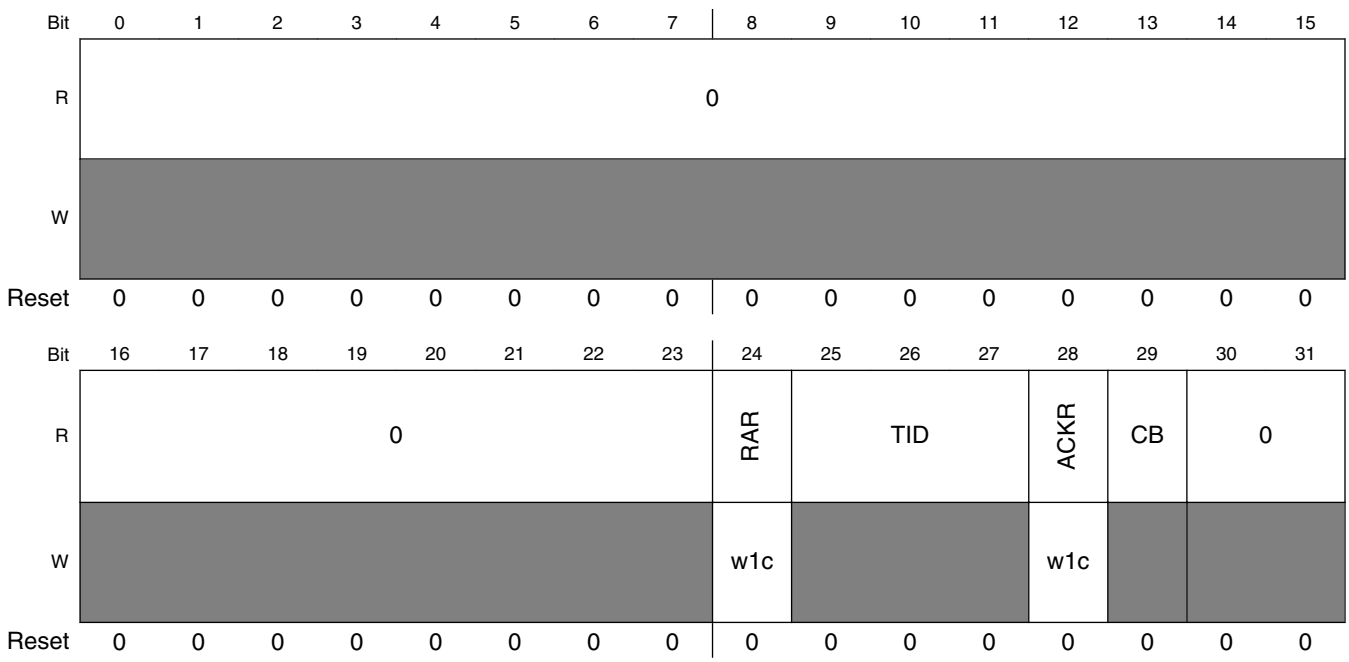
SIPI_CCR1 field descriptions (continued)

Field	Description
	0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1 or SIPI_CCR1[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

55.15.9 SIPI Channel Status Register 1 (SIPI_CSR1)

CSR1 contains the status bits for the current transfer.

Address: 0h base + 24h offset = 24h



SIPI_CSR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception.

Table continues on the next page...

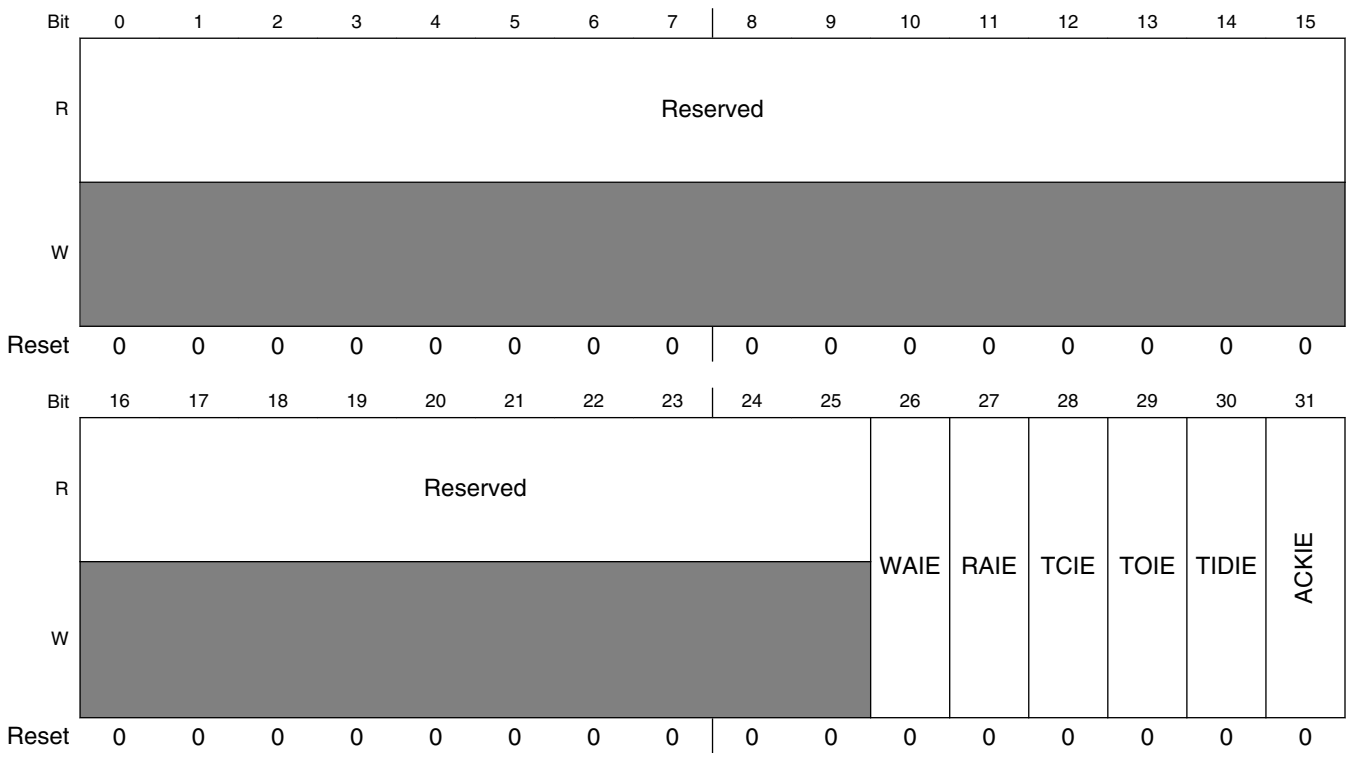
SIPI_CSR1 field descriptions (continued)

Field	Description
	0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 1 status. 0 Channel 1 free 1 Channel 1 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.15.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)

The CIR1 contains the interrupt enable bits for channel 1.

Address: 0h base + 2Ch offset = 2Ch



SIPI_CIR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR1 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

55.15.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)

SIPI_CTOR1 contains the timeout value for Tx requests.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

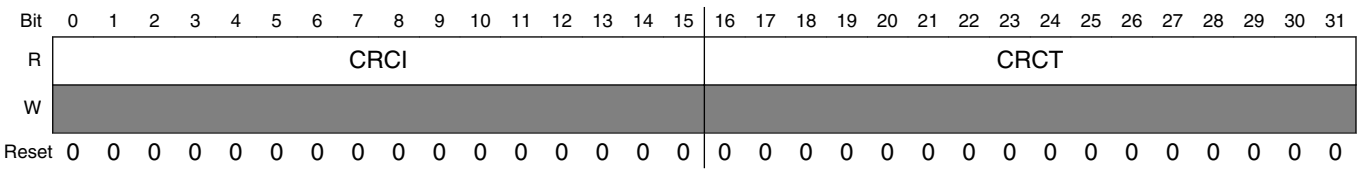
SIPI_CTOR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

55.15.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 34h offset = 34h



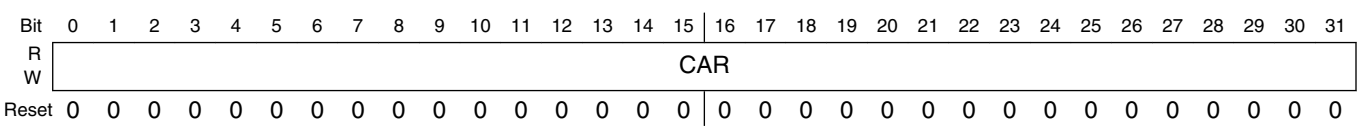
SIPI_CCRC1 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

55.15.13 SIPI Channel Address Register 1 (SIPI_CAR1)

SIPI_CAR1 is the address target for data transmission.

Address: 0h base + 38h offset = 38h



SIPI_CAR1 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

55.15.14 SIPI Channel Data Register 1 (SIPI_CDR1)

SIPI_CDR1 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR1 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

55.15.15 SIPI Channel Control Register 2 (SIPI_CCR2)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR2[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR2[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 2 (SIPI_CSR2)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 2 (SIPI_CAR2)).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR2 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR2. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
	0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1 or SIPI_CCR2[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

55.15.16 SIPI Channel Status Register 2 (SIPI_CSR2)

SIPI_CSR2 contains the status bits for the current transfer.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RAR	TID			ACKR	CB	0	
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception.

Table continues on the next page...

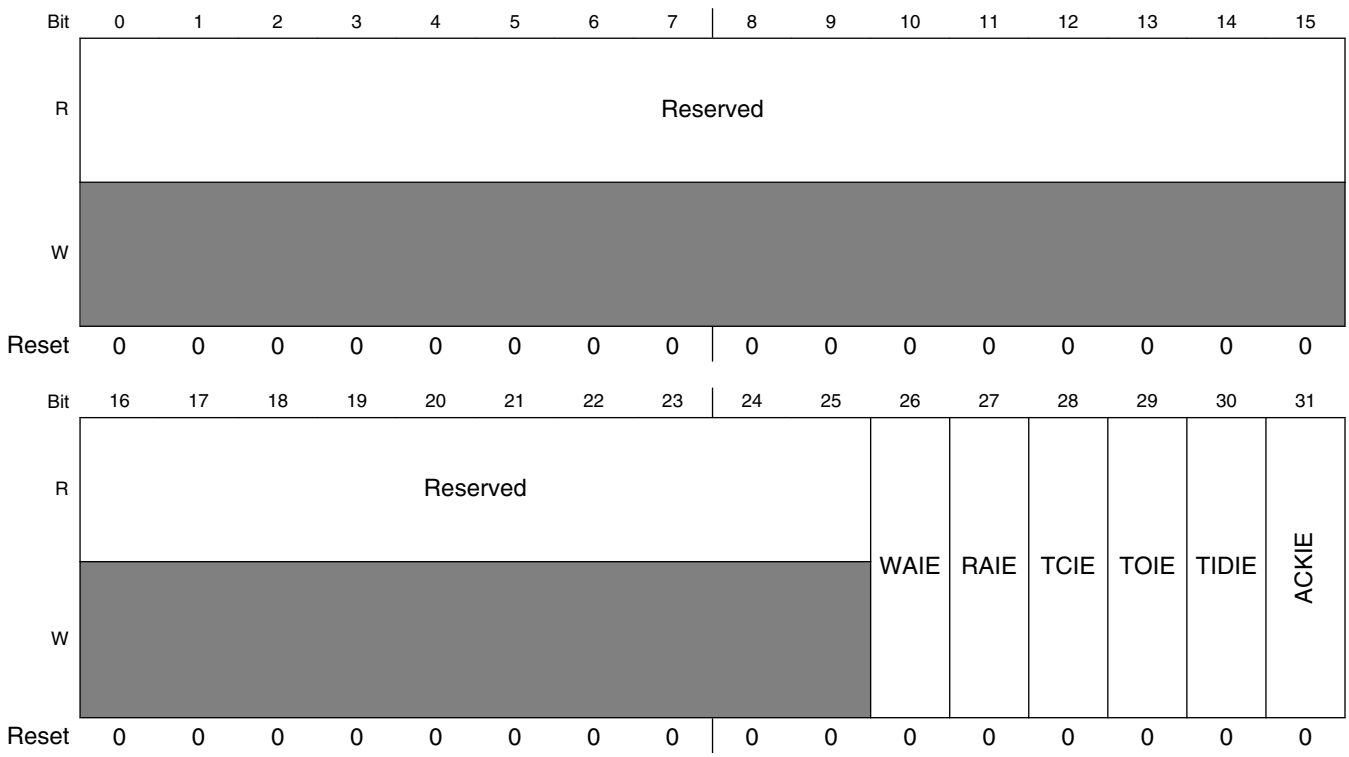
SIPI_CSR2 field descriptions (continued)

Field	Description
	0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 2 status. 0 Channel 2 free 1 Channel 2 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.15.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)

The SIPI_CIR2 contains the interrupt enable bits for channel 2.

Address: 0h base + 4Ch offset = 4Ch



SIPI_CIR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR2 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

55.15.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)

SIPI_CTOR2 contains the timeout value for Tx requests.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

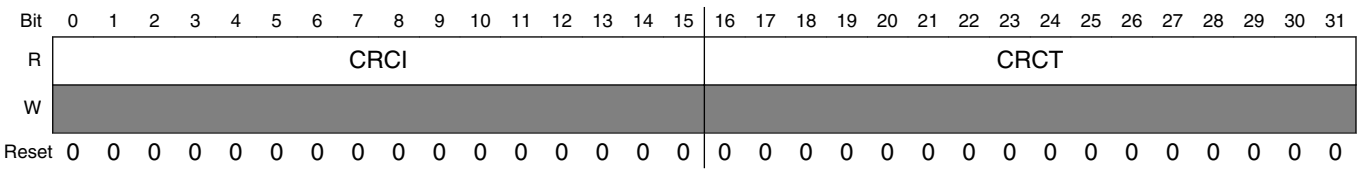
SIPI_CTOR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

55.15.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 54h offset = 54h



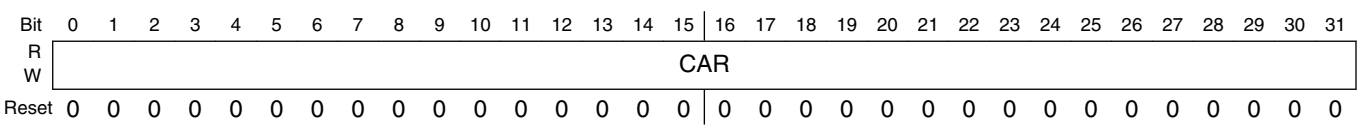
SIPI_CCRC2 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

55.15.20 SIPI Channel Address Register 2 (SIPI_CAR2)

SIPI_CAR2 is the address target for data transmission. For streaming operations this register is the start address.

Address: 0h base + 58h offset = 58h



SIPI_CAR2 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

55.15.21 SIPI Channel Data Register 2 (SIPI_CDR2_*n*)

SIPI_CDR2_*n*contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 5Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR2_*n* field descriptions

Field	Description
0–31 CDR2	Data register bits. Contains the data that will be transmitted, or received.

55.15.22 SIPI Channel Control Register 3 (SIPI_CCR3)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR3[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR3[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 3 (SIPI_CSR3)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 3 (SIPI_CAR3)).

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR3 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR3. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR3 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT

Table continues on the next page...

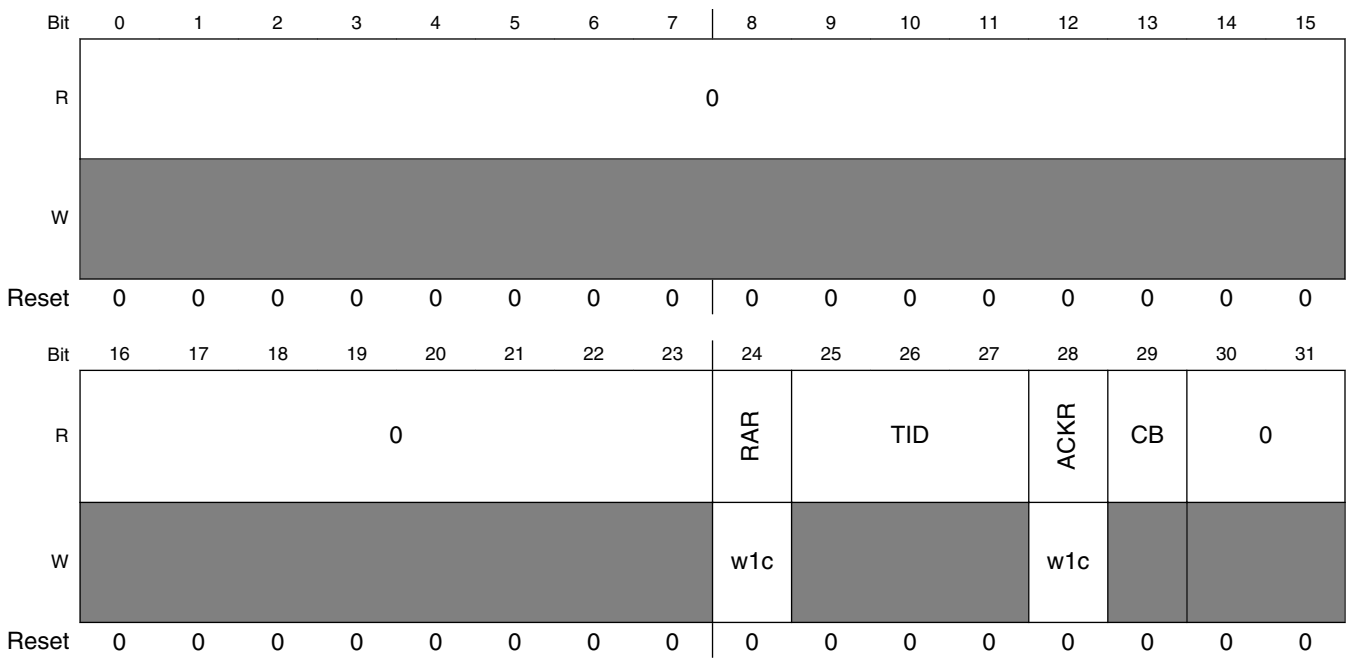
SIPI_CCR3 field descriptions (continued)

Field	Description
	3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1 or SIPI_CCR3[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

55.15.23 SIPI Channel Status Register 3 (SIPI_CSR3)

SIPI_CSR3 contains the status bits for the current transfer.

Address: 0h base + 80h offset = 80h



SIPI_CSR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

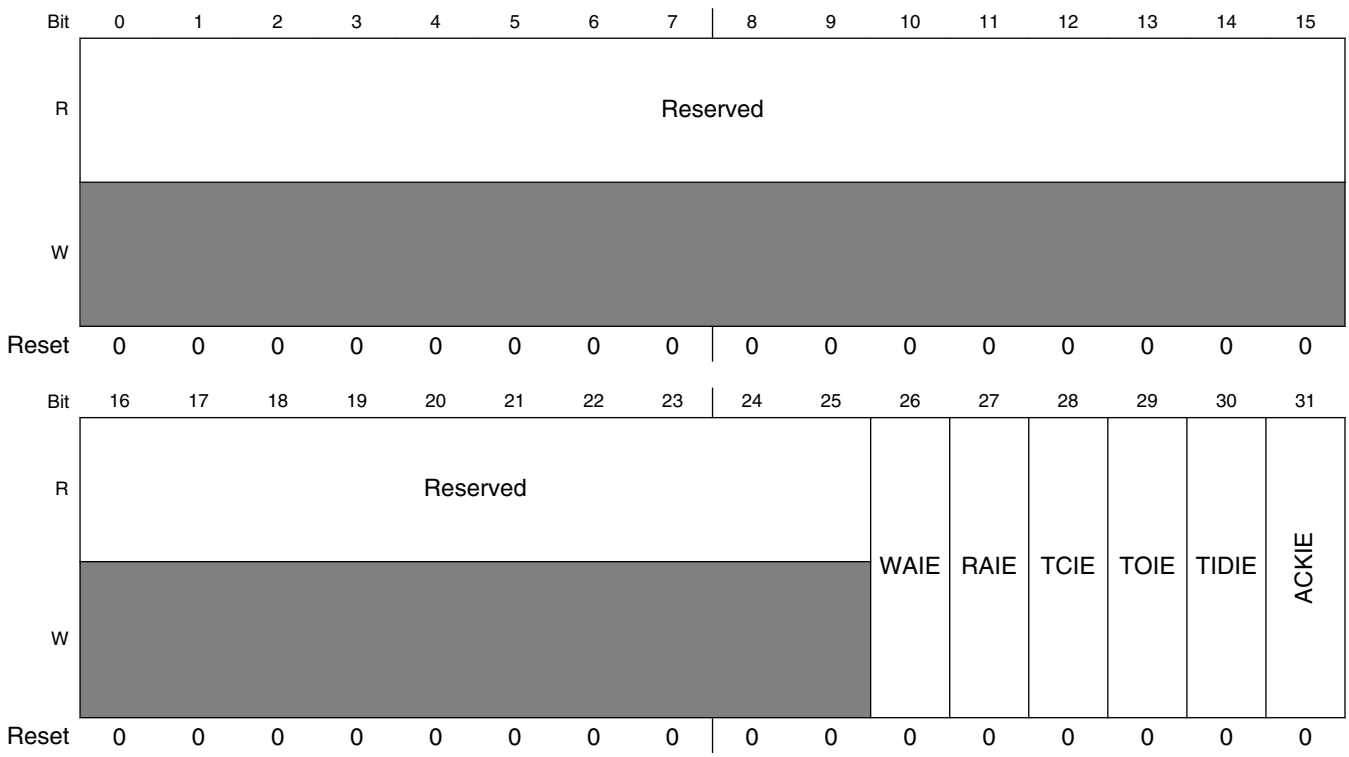
SIPI_CSR3 field descriptions (continued)

Field	Description
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 3 status. 0 Channel 3 free 1 Channel 3 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

55.15.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)

SIPI_CIR3 contains the interrupt enable bits for channel 3.

Address: 0h base + 88h offset = 88h



SIPI_CIR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR3 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

55.15.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)

SIPI_CTOR3 contains the timeout value for Tx requests.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

55.15.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC3 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

55.15.27 SIPI Channel Address Register 3 (SIPI_CAR3)

SIPI_CAR3 is the address target for data transmission.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR3 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

55.15.28 SIPI Channel Data Register 3 (SIPI_CDR3)

SIPI_CDR3 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR3 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

55.15.29 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FRZ	Reserved	HALT	0		PRSCLR										
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AID		0			CRCIE	MCRIE	0				CHNSB	TEN	INIT	MOEN	SR
Reset	0*	0*	0	0	0	0*	0*	0	0	0	0	0*	0*	0*	0	0

* Notes:

- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.
- TEN field: Can only be written when SIPI_MCR[MOEN] = 1.
- CHNSB field: Can be written only once after reset.
- MCRIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- CRCIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- AID field: Can be written in initialization mode only (SIPI_MCR[INIT] = 1).

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved. **Important:** Always write the default value to this field.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 PRSCLR	Timeout counter prescaler The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are (all other values are ignored): 040h 64 (default) 080h 128 100h 256 200h 512 400h 1024 **NOTE:** This field should be programmed by software during initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. **NOTE:** Writes to SIPI_MCR[PRSCLR] can only be accomplished with 16-bit or 32-bit writes.
16–17 AID	Address Increment/Decrement These bits define the type of address change at the target node. **NOTE:** This bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4 11 not used
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

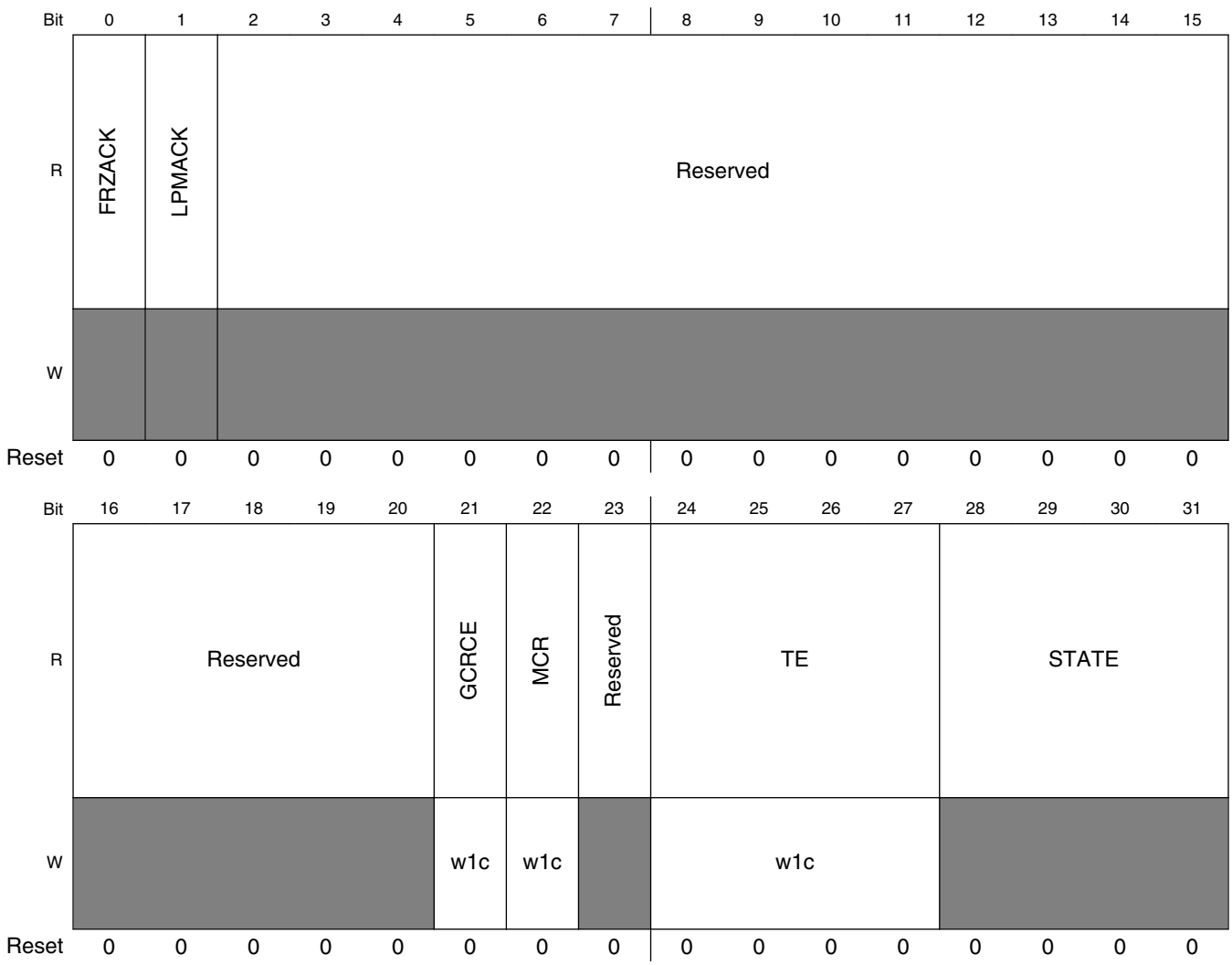
SIPI_MCR field descriptions (continued)

Field	Description
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
23–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 CHNSB	Channel coding select bit. 0 Code Table II (see Table 55-2) 1 Code Table I (see Table 55-2)
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in "streaming without ACK" mode. This bit has to be enabled for the transmission operations also.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

55.15.30 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h



SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited.

Table continues on the next page...

SIPI_SR field descriptions (continued)

Field	Description
	0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge. This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI is not in low power mode 1 SIPI is in Disable Mode.
2–20 Reserved	This field is reserved.
21 GCRCE	Global CRC Error Bit. 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This it should be cleared by software.
23 Reserved	This field is reserved.
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. xxx1 TE0 = 1 - Channel 0 trigger event xx1x TE1 = 1 - Channel 1 trigger event x1xx TE2 = 1 - Channel 2 trigger event 1xxx TE3 = 1 - Channel 3 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

55.15.31 SIPI Max Count Register (SIPI_MAXCR)

SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MXCNT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MXCNT														Reserved	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30–31 Reserved	This field is reserved.

55.15.32 SIPI Address Reload Register (SIPI_ARR)

The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

NOTE

This register is writeable only when SIPI_MCR[INIT] = 1.

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADRLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADRLD														Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	ADRLD contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31 Reserved	This field is reserved.

55.15.33 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

NOTE

SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

NOTE

When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR (address count register).

NOTE

After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4 or remain the same (depending on configuration). In both cases, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADCNT														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Feflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30–31 Reserved	This field is reserved.

55.15.34 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					TOE3	TIDE3	ACKE3	0					TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					TOE1	TIDE1	ACKE1	0					TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error didn't occur 1 Timeout error occured
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error didn't occur 1 Timeout error occured
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error didn't occur 1 Timeout error occured
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error didn't occur 1 Timeout error occured

Table continues on the next page...

SIPI_ERR field descriptions (continued)

Field	Description
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

Chapter 56
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

56.1 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

56.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

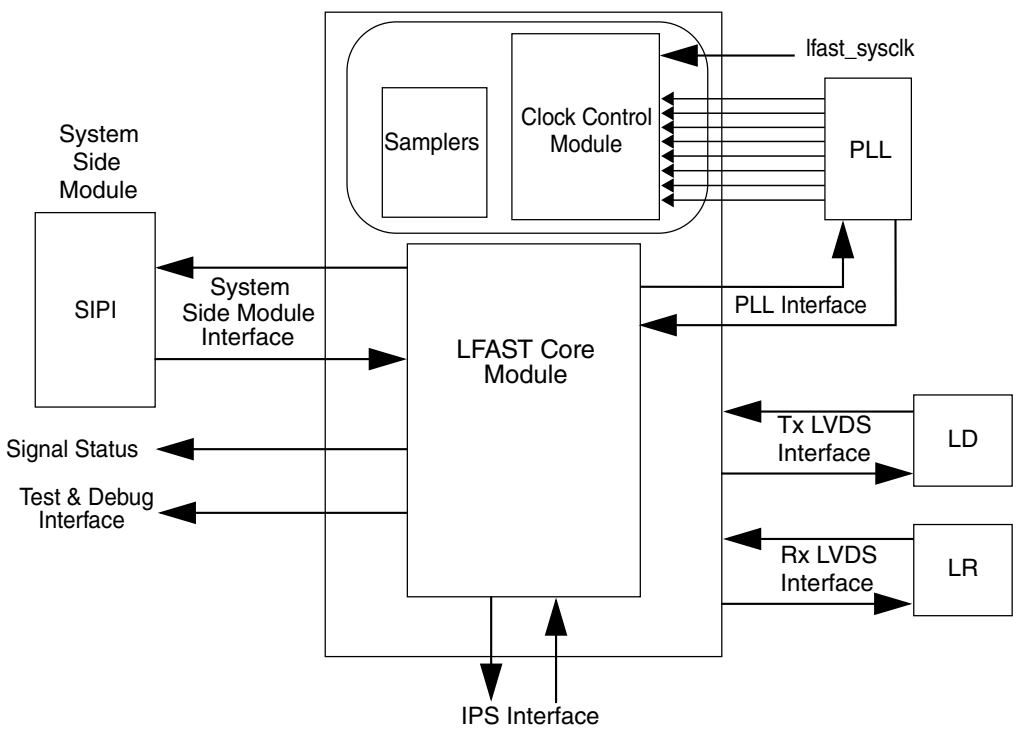


Figure 56-1. LFAST block diagram

56.3 External signals

LFAST is a five pin interface with the following signals :

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

56.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

56.4 LFAST frame structure

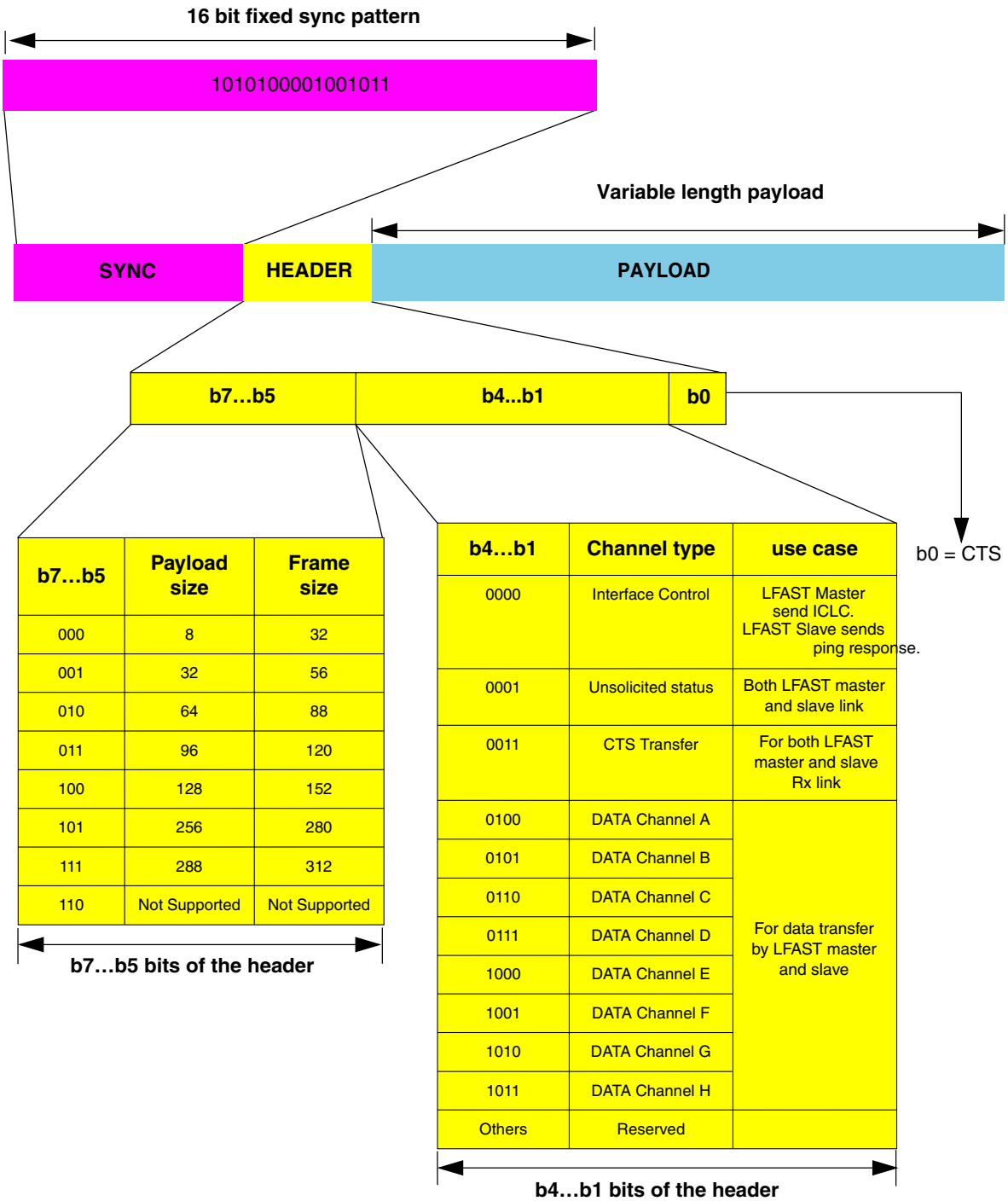
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 56-2 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312
110	Not Supported	Not Supported

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 56-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

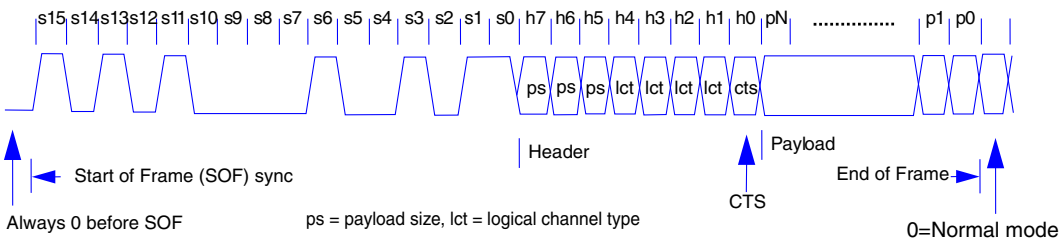


Figure 56-3. Serial frame structure

The frame structure is shown in Figure 56-3 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 56-1 :

Table 56-1. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.

- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

56.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer up to the maximum data rate shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Receives ICLC frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

56.6 Memory map and register definition

All registers are 32 bits wide.

NOTE

Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	56.6.1/2592
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	56.6.2/2594
8	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	56.6.3/2595
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	56.6.4/2597
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	56.6.5/2598
14	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	56.6.6/2599
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	56.6.7/2599
1C	LFAST ICLC Control Register (LFAST_ICR)	32	R/W	0000_0000h	56.6.8/2601
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	56.6.9/2602
2C	LFAST Rx FIFO CTS Control Register (LFAST_RFCR)	32	R/W	000F_0009h	56.6.10/ 2603
30	LFAST Tx Interrupt Enable Register (LFAST_TIER)	32	R/W	0000_0000h	56.6.11/ 2603
34	LFAST Rx Interrupt Enable Register (LFAST_RIER)	32	R/W	0000_0000h	56.6.12/ 2604
38	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)	32	R/W	0000_0000h	56.6.13/ 2606
3C	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	56.6.14/ 2608
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	56.6.15/ 2610
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	56.6.16/ 2613
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	56.6.17/ 2613
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	56.6.17/ 2613
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	56.6.17/ 2613
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	56.6.17/ 2613

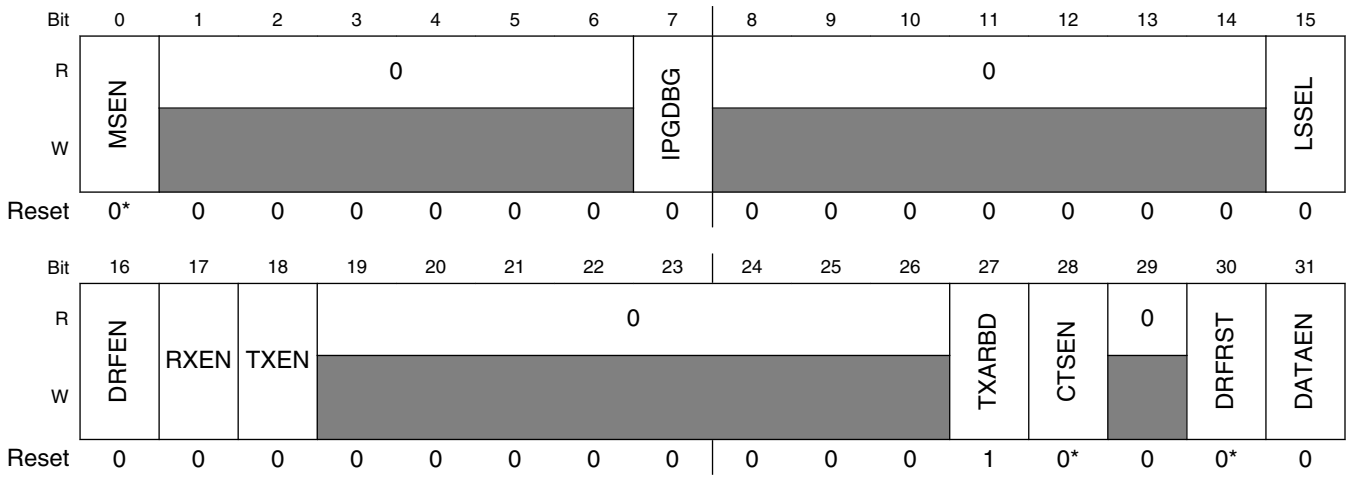
Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	56.6.17/ 2613
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	56.6.17/ 2613
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	56.6.17/ 2613
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	56.6.17/ 2613
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	56.6.17/ 2613
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	56.6.18/ 2614
84	LFAST Ping Status Register (LFAST_PISR)	32	R	0000_0000h	56.6.19/ 2615
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	56.6.20/ 2616
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	56.6.21/ 2617
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	56.6.22/ 2618
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	w1c	0000_0000h	56.6.23/ 2620
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	56.6.24/ 2622
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	56.6.25/ 2623
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	56.6.26/ 2624
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	56.6.26/ 2624
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	56.6.26/ 2624
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	56.6.26/ 2624
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	56.6.26/ 2624
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	56.6.26/ 2624
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	56.6.26/ 2624
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	56.6.26/ 2624
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	56.6.26/ 2624

56.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 0h base + 0h offset = 0h



* Notes:
- DRFRST field: Set by user software and then cleared by system hardware.
- CTSEN field: Only writable when MCR[DRFEN] = 0.
- MSEN field: Writable only once after the asynchronous reset.

LFAST_MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable

Table continues on the next page...

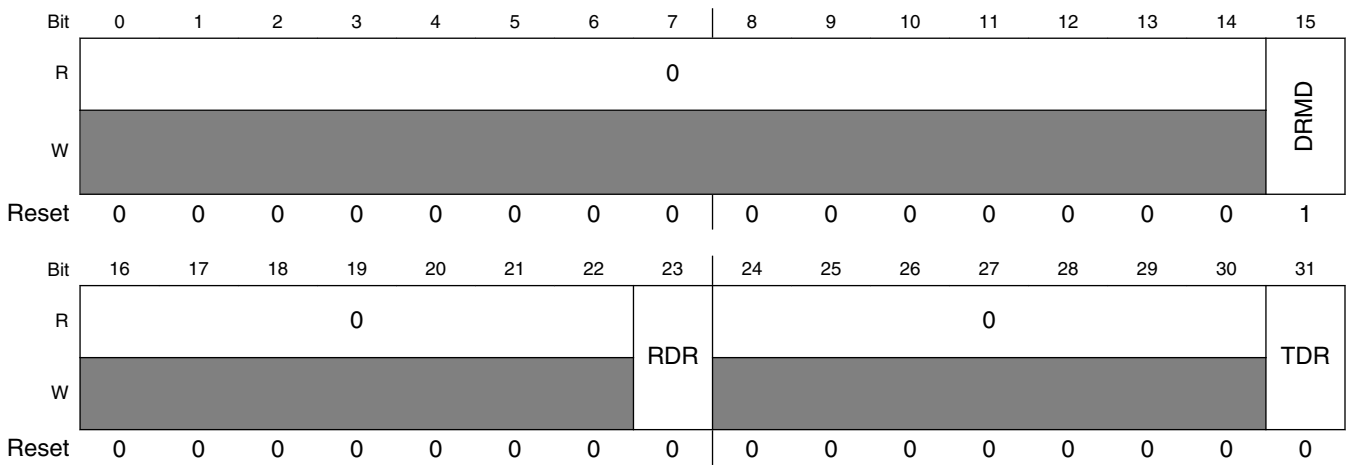
LFAST_MCR field descriptions (continued)

Field	Description
	registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be modified by LFAST slave H/W on reception of an ICLC command frame. This field can only be written in dual mode (LFAST_GSR[DUALMD] = 1). 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 LFAST transmitter Interface is enabled. New requests are accepted.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. This bit is only writable when MCR[DRFEN] = 0. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit doesn't represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

56.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 0h base + 4h offset = 4h



LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no affect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.

Table continues on the next page...

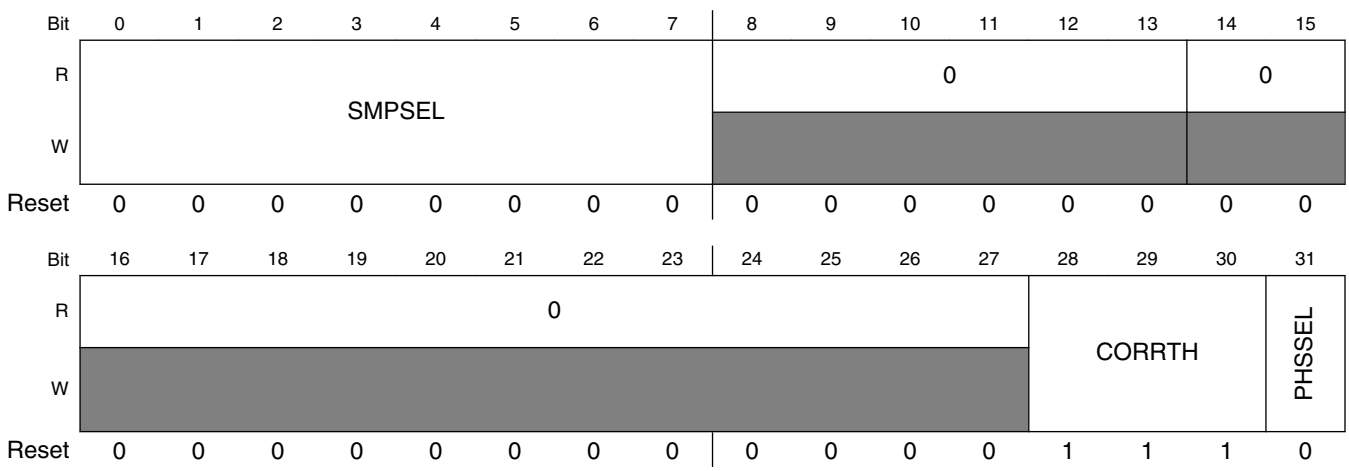
LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Tx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

56.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 0h base + 8h offset = 8h



LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0.

Table continues on the next page...

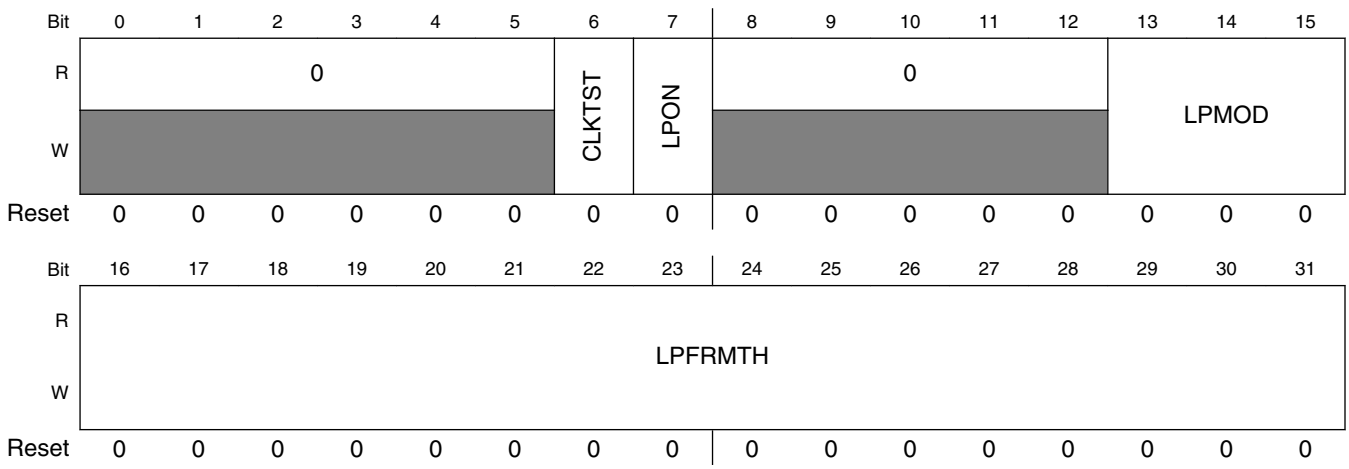
LFAST_COCR field descriptions (continued)

Field	Description
	00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

56.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 0h base + Ch offset = Ch



LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

56.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LPCNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
	... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

56.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
W																																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

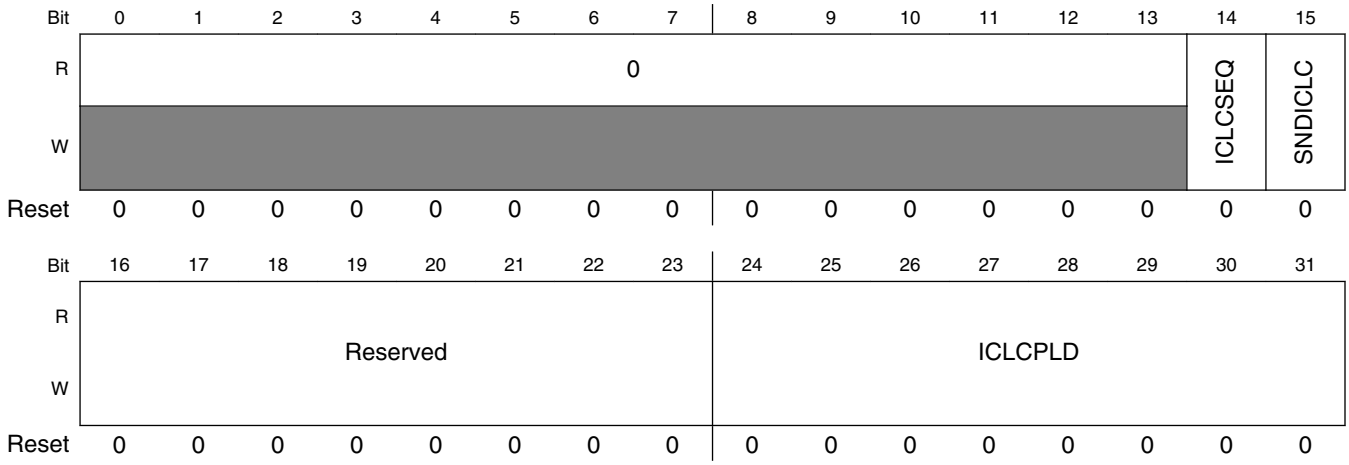
Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

56.6.8 LFAST ICLC Control Register (LFAST_ICR)

Address: 0h base + 1Ch offset = 1Ch

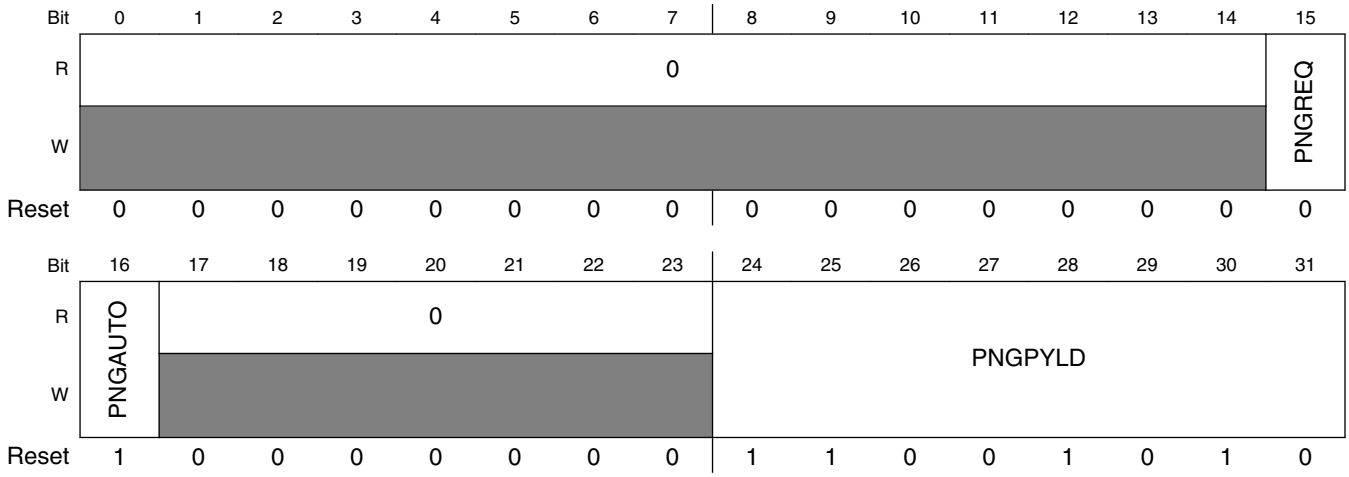


LFAST_ICR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 0 Single ICLC frame transfer. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0.
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. Set by user software and cleared by system hardware. 0 No Valid ICLC frame for transfer. 1 Valid ICLC frame for transfer.
16–23 Reserved	This field is reserved.
24–31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. This field can only be written when ICR[SNDICLC] = 0 (see Table 56-8 for supported ICLC payloads).

56.6.9 LFAST Ping Control Register (LFAST_PICR)

Address: 0h base + 20h offset = 20h



LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. This field can only be written when MCR[DRFEN] = 1. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

56.6.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)

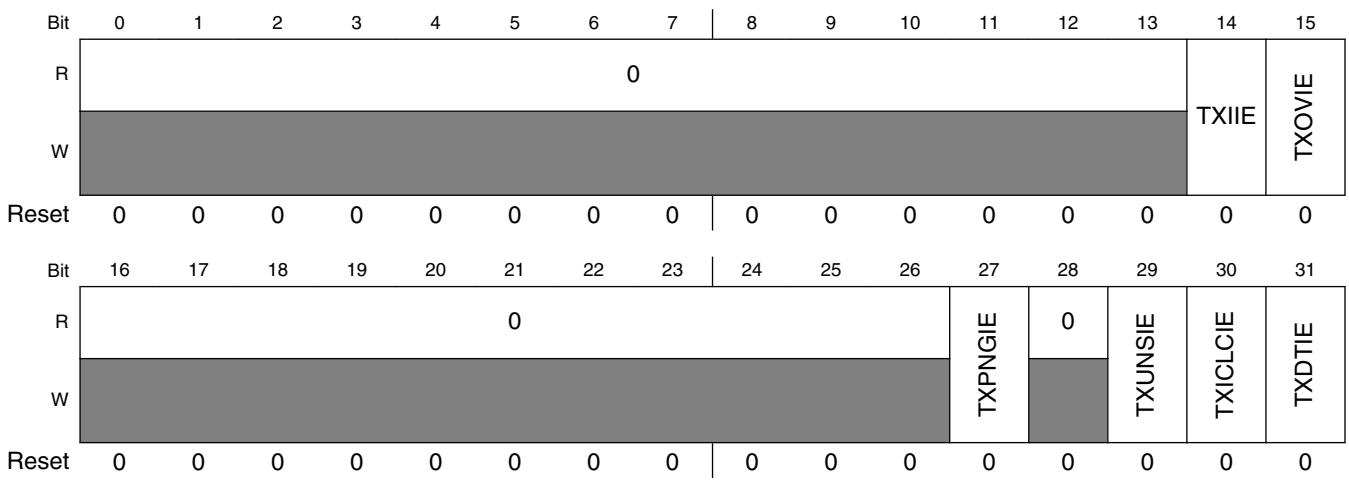
Address: 0h base + 2Ch offset = 2Ch

LFAST_RFCR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated. This field can only be written when LFAST_MCR[DRFEN] = 0.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set. This field can only be written when LFAST_MCR[DRFEN] = 0.

56.6.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)

Address: 0h base + 30h offset = 30h



LFAST_TIER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIIE	Tx Data Interface Not Enabled - (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted

Table continues on the next page...

LFAST_TIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

56.6.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIER field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 56-12 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled

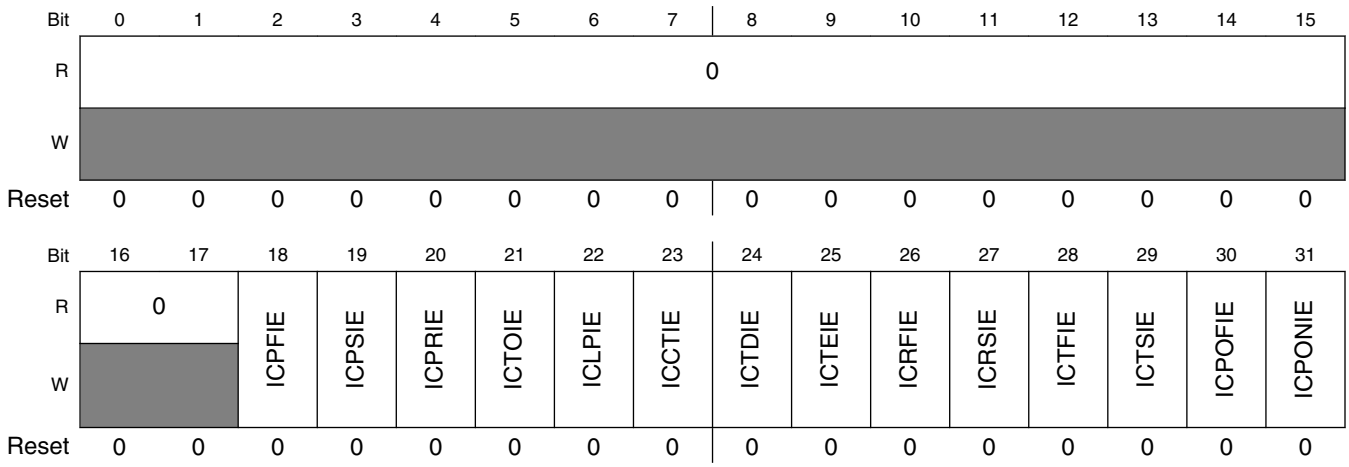
Table continues on the next page...

LFAST_RIER field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.6.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)

Address: 0h base + 38h offset = 38h



LFAST_RIIER field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled

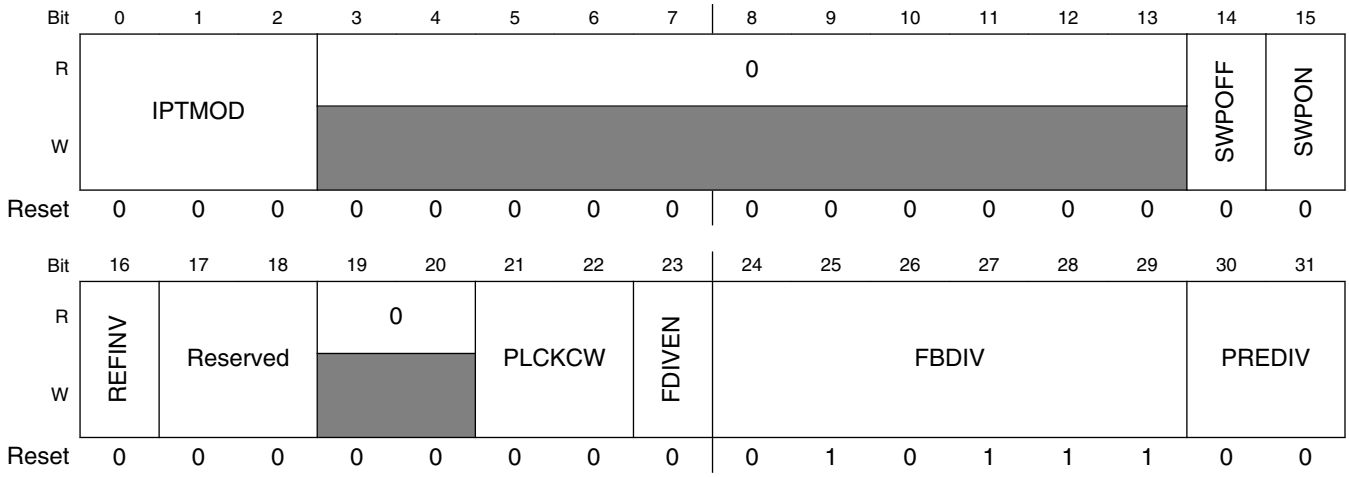
Table continues on the next page...

LFAST_RIIER field descriptions (continued)

Field	Description
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

56.6.14 LFAST PLL Control Register (LFAST_PLLCR)

Address: 0h base + 3Ch offset = 3Ch



LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	SW signal to turn OFF the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert

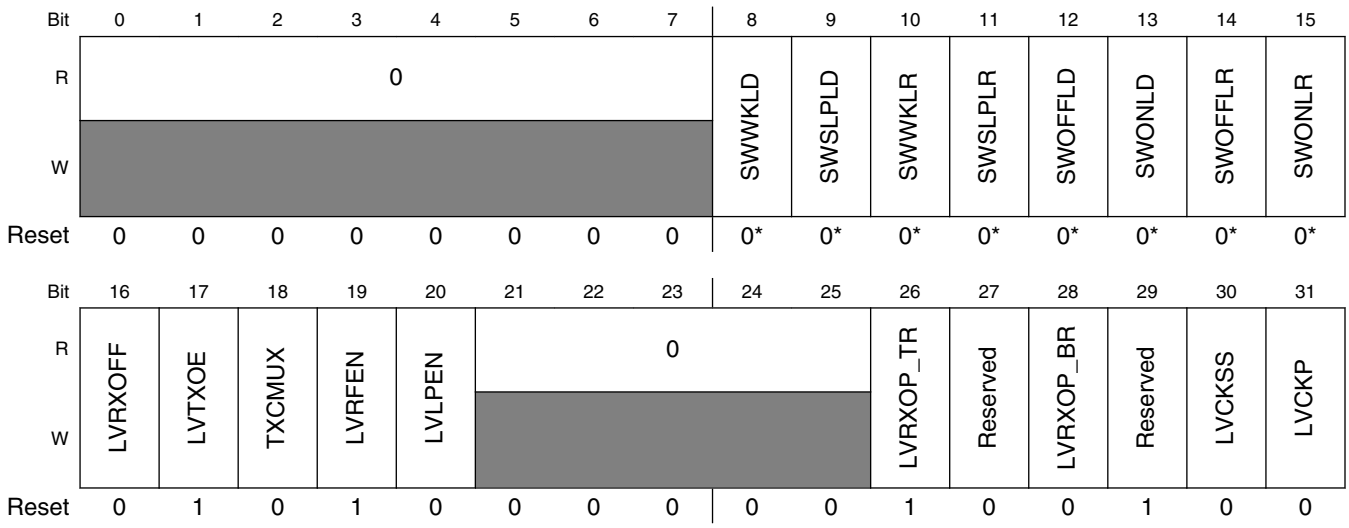
Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
17–18 Reserved	This field is reserved.
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved 0Bh Divide by 12 (11.5, if FDIVEN = 1) ... ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

56.6.15 LFAST LVDS Control Register (LFAST_LCR)

Address: 0h base + 40h offset = 40h



* Notes:
- SWONLR field: Set by user software and cleared by system hardware.
- SWOFFLR field: Set by user software and cleared by system hardware.
- SWONLD field: Set by user software and cleared by system hardware.
- SWOFFLD field: Set by user software and cleared by system hardware.
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled

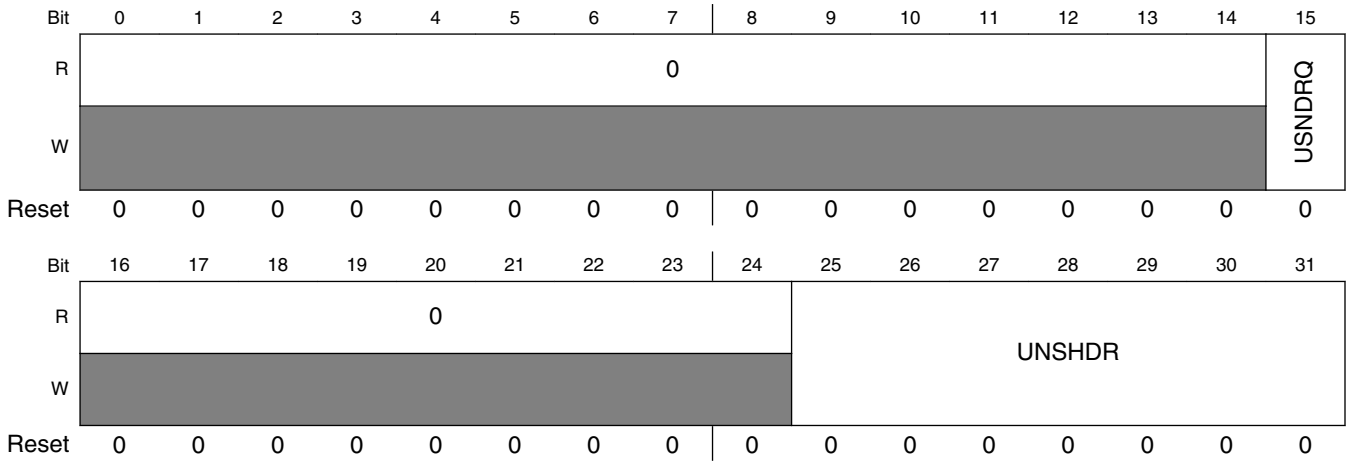
Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal ipp_digrf_lvds_lpbk_en.The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 1 Disable on-chip LFAST receiver termination 0 Enable on-chip LFAST receiver termination
27 Reserved	This field is reserved.
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than the maximum baud rate. 1 Required for LFAST receiver maximum baud rate.
29 Reserved	This field is reserved.
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

56.6.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 0h base + 44h offset = 44h

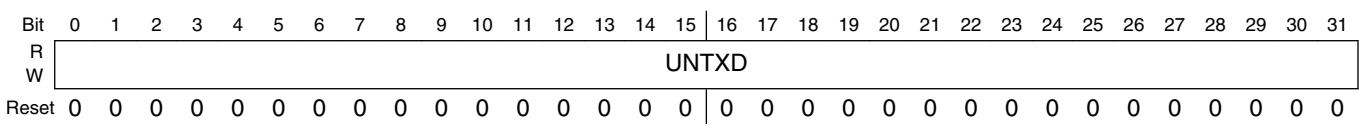


LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. This field can only be written when MCR[DRFEN] = 1. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

56.6.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 0h base + 48h offset + (4d × i), where i=0d to 8d

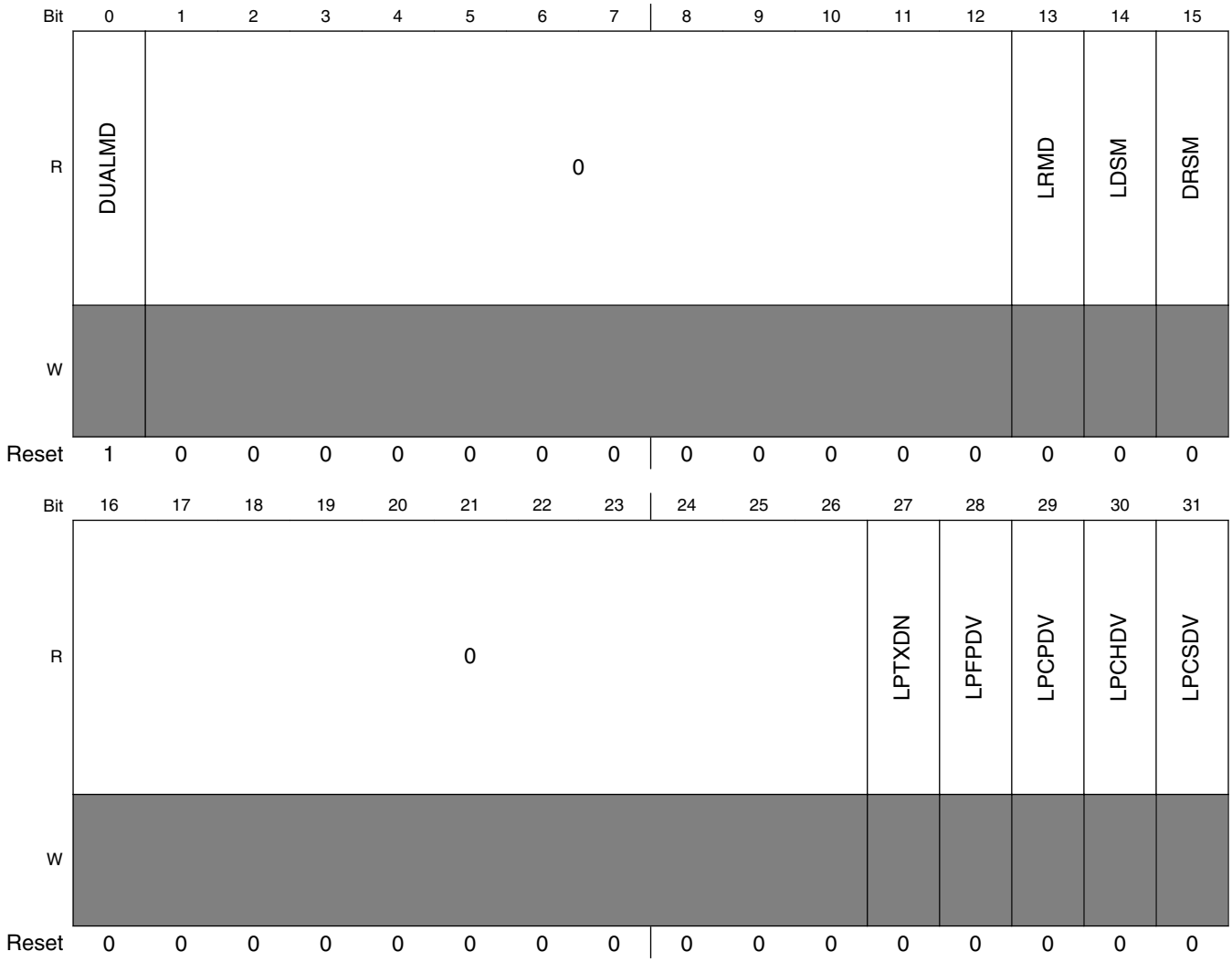


LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

56.6.18 LFAST Global Status Register (LFAST_GSR)

Address: 0h base + 80h offset = 80h



LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

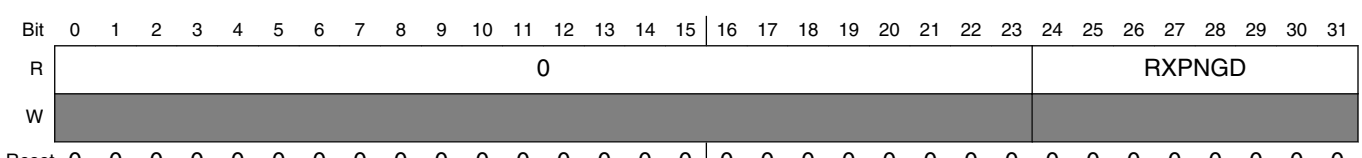
Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

56.6.19 LFAST Ping Status Register (LFAST_PISR)

Address: 0h base + 84h offset = 84h



LFAST_PISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

56.6.20 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 0h base + 94h offset = 94h

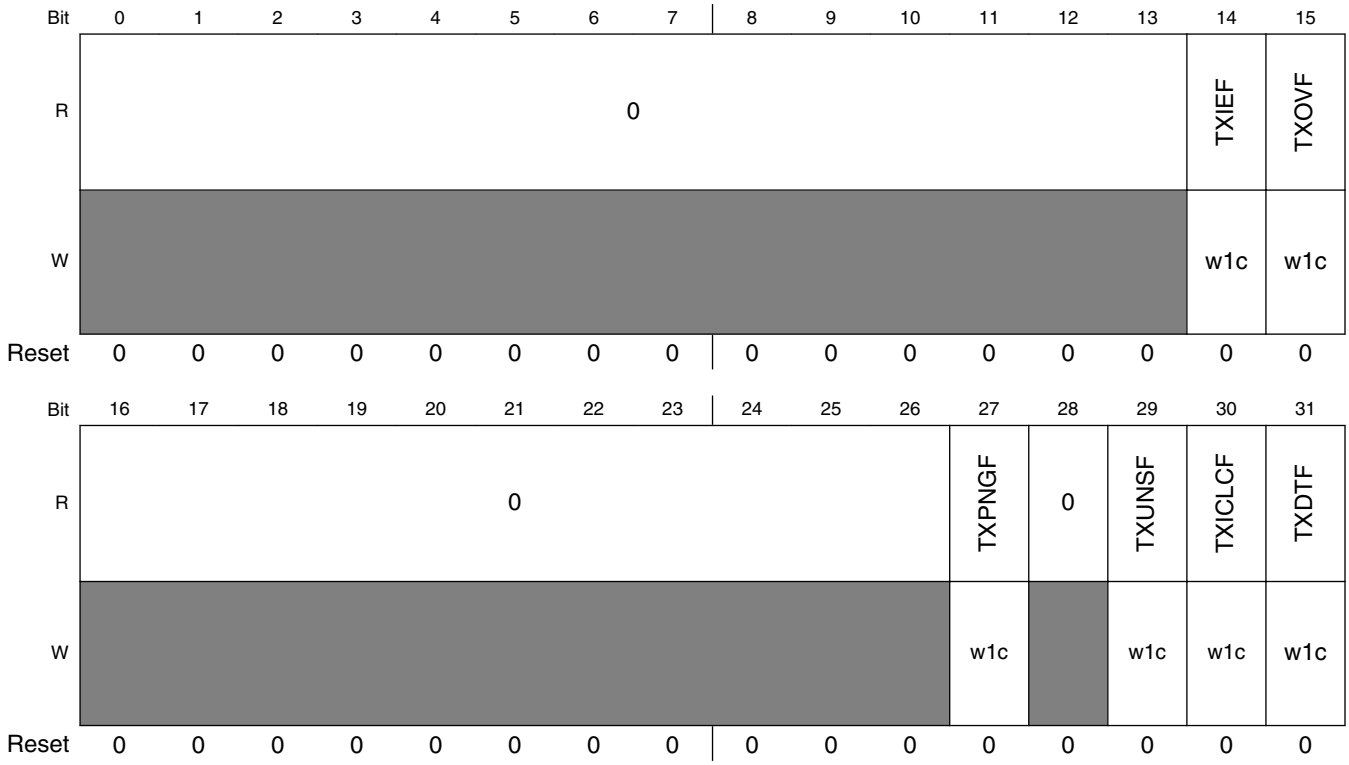
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT						0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

56.6.21 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 0h base + 98h offset = 98h



LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_TISR field descriptions (continued)

Field	Description
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

56.6.22 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.6.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received

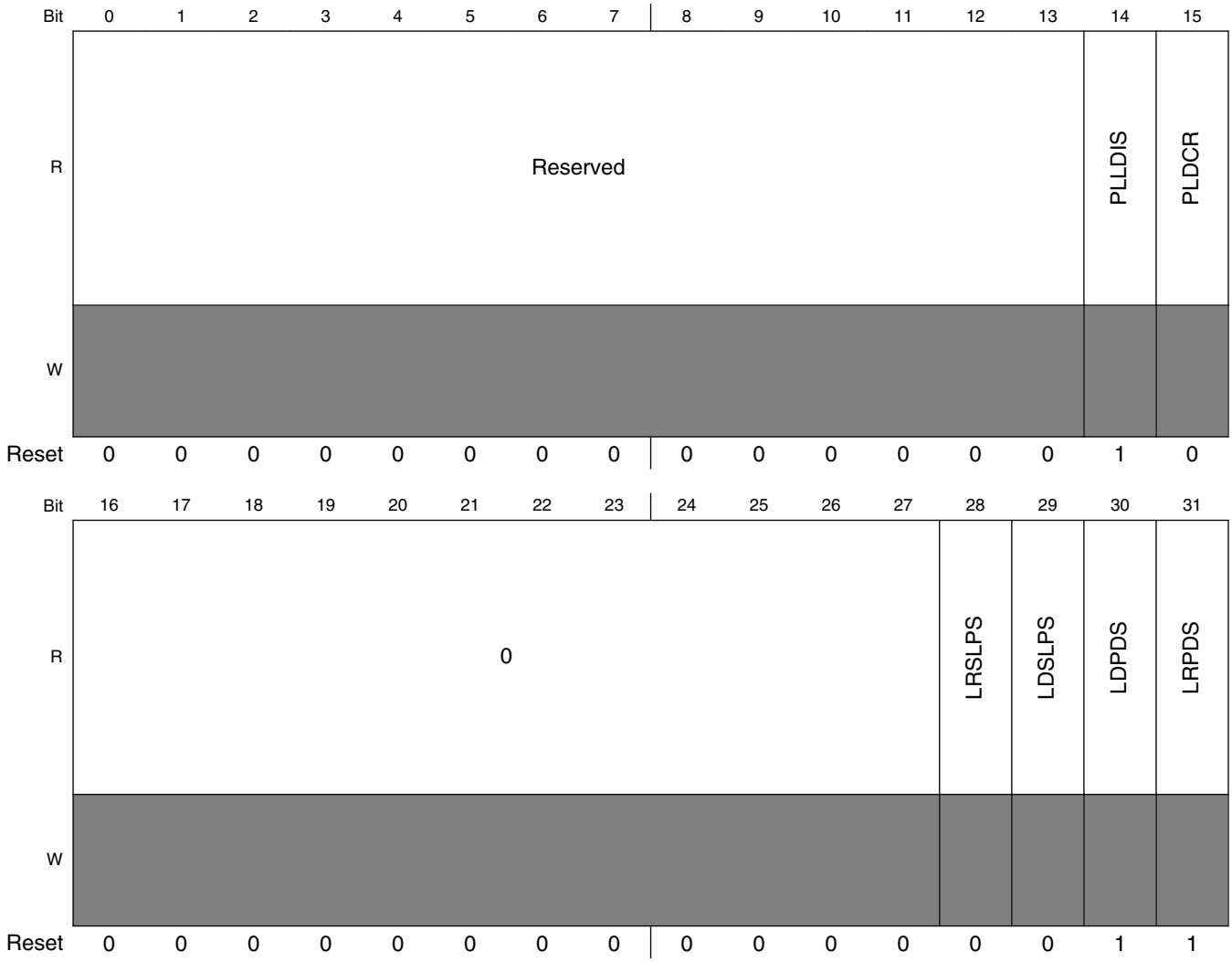
Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

56.6.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 0h base + A4h offset = A4h



LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

56.6.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers. **NOTE:** Writing 0 will clear this bit
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

56.6.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 0h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

56.7 Functional description

56.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

56.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS parameters in the device data sheet.

2. The PLLCR is programmed with configuration parameters for the PLL.
3. The LCR is programed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN]and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
    - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
    - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write PLLCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:

Note

(The slaves Rx interface speed should be changed first.)

- Write ICR[ICLCPLD] = 10h for Tx data fast frame.
- Write ICR[SNDICLC] = 1.

19. Both of the following operations are performed to change the LFAST masters Rx interface speed:

Note

(The slaves Tx interface speed mode should be changed first)

- Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
- Write ICR[SNDICLC] = 1.

20. The ICLC transmission is confirmed by the occurrence of one of the following:
    - ICR[SNDICLC] = 0
    - TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
24. Write ICR[SNDICLC] = 1
25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitation.
26. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    - If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
    - If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:

- Write ICR[ICLCSEQ] = 0.
- Write MCR[TXARBD] = 0.

28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    - Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    - Write MCR[TXARBD] = 0, to enable frame transmission.
    - The operations from step 14 need to be executed again.

56.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signaland LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
    - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
    - MCRMCR[TXEN] = 1.
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received. H/W writes [ICRPF] = 1.
    - PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:

- When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1.
- Write PLLCR[SWPON] = 1.

10. The speed of the Tx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
    - SCR[TDR] = 1, when SCR[DRMD] = 0.
11. The speed of the Rx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
    - SCR[RDR] = 1, when SCR[DRMD] = 0.
12. A Ping Frame is sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received. H/W writes RIISR[ICRPF] = 1.
    - Write PICR[PNGREQ] = 1.
    - Write PICR[PNGREQ] = .

56.7.2 Line Receiver

This section describes the Line Receiver.

56.7.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data

type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 56-4 shows the block diagram of Uplink Controller.

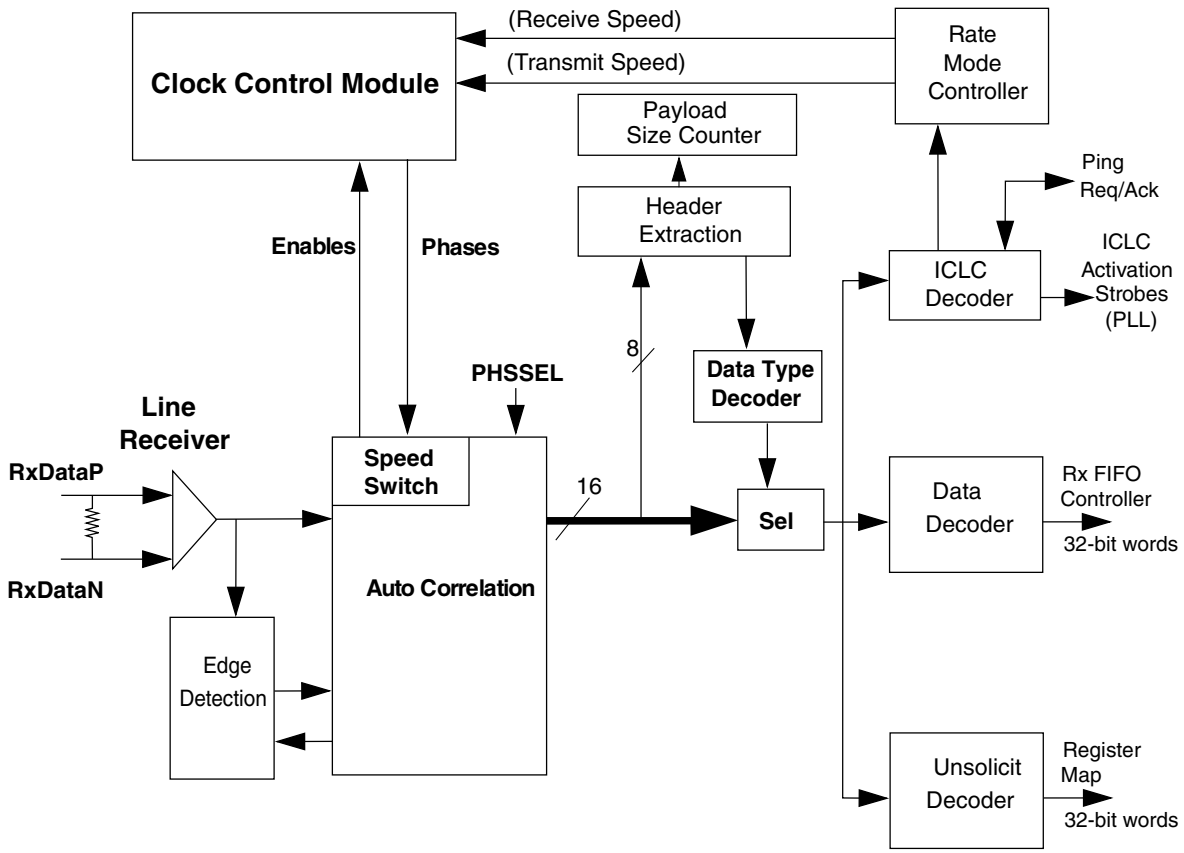


Figure 56-4. Top Level Receive Controller

56.7.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

56.7.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

56.7.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 56-5 shows the block diagram of Hunt Correlation mode.

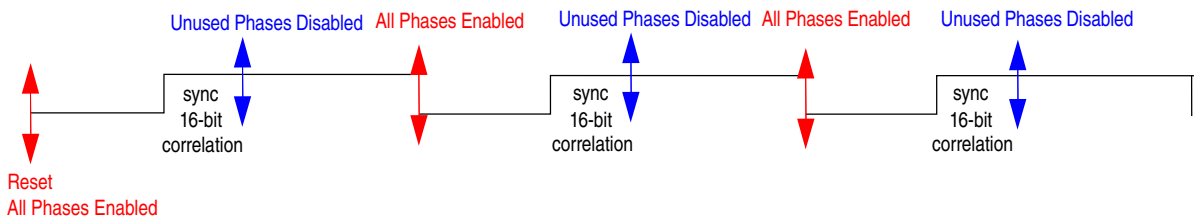


Figure 56-5. Hunt Correlation mode

56.7.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

56.7.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

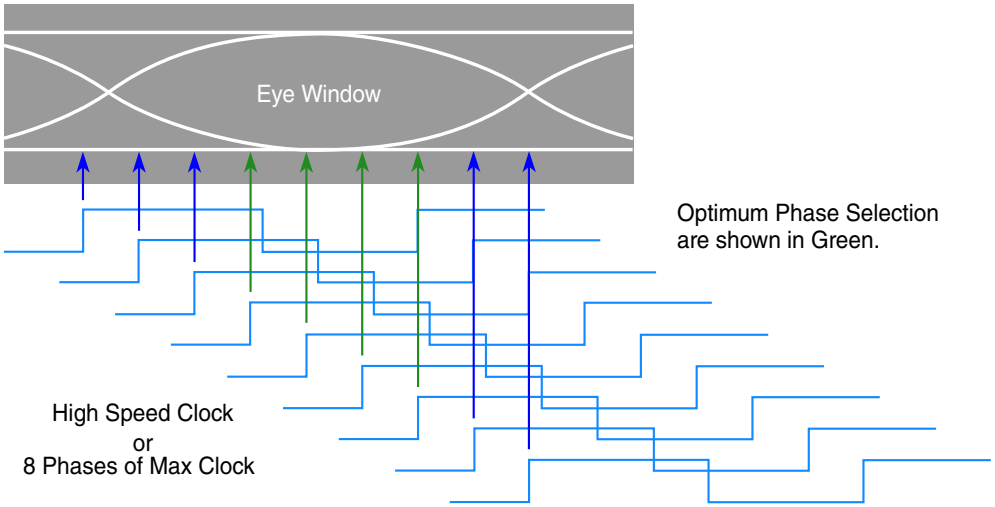


Figure 56-6. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

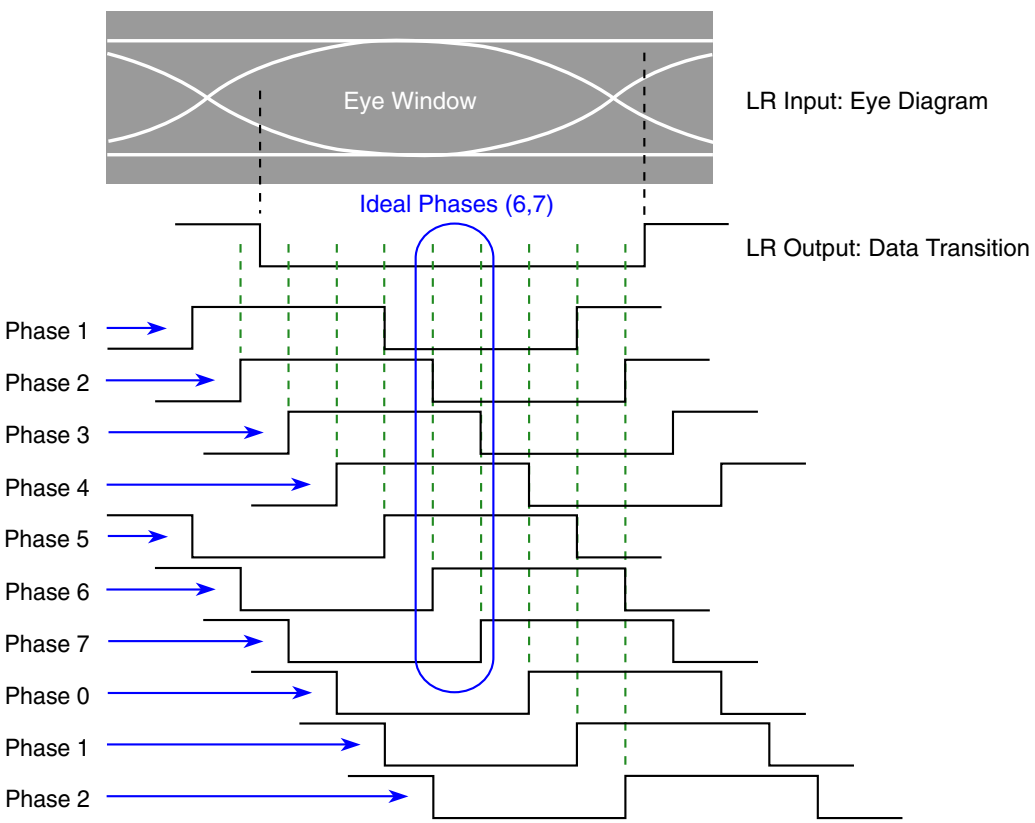


Figure 56-7. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 56-8 and Figure 56-9.

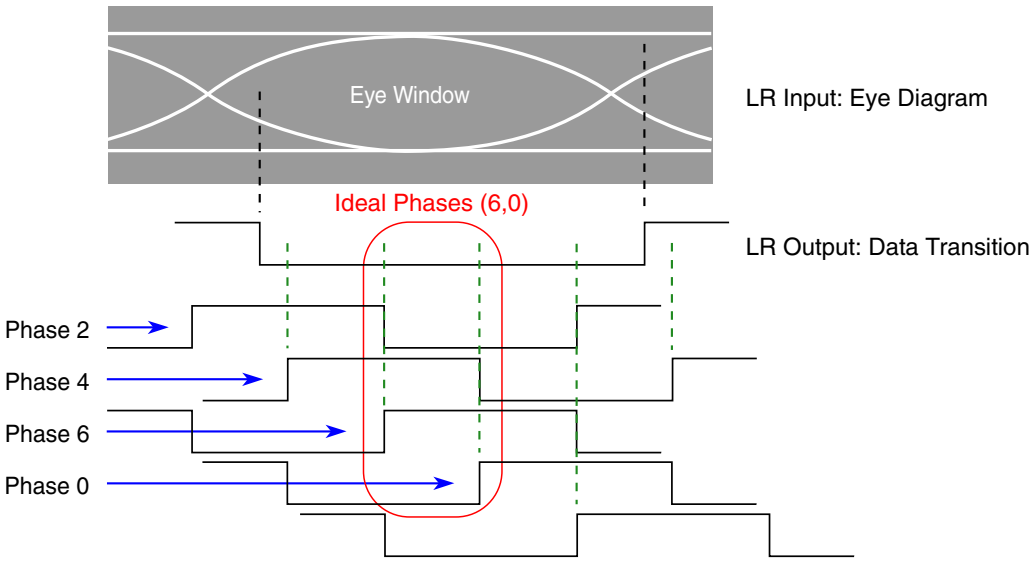


Figure 56-8. High Speed 4 Phase Clock Alignment Example

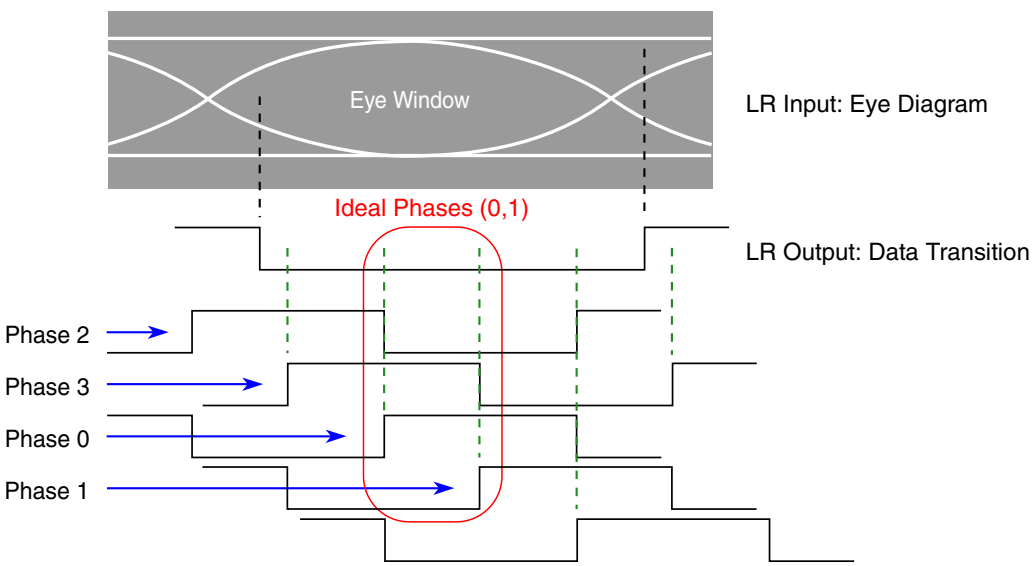


Figure 56-9. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 56-2, Table 56-3, and Table 56-4 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 56-2. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 56-3. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 56-3, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 56-4. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 56-4. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 56-4, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

56.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

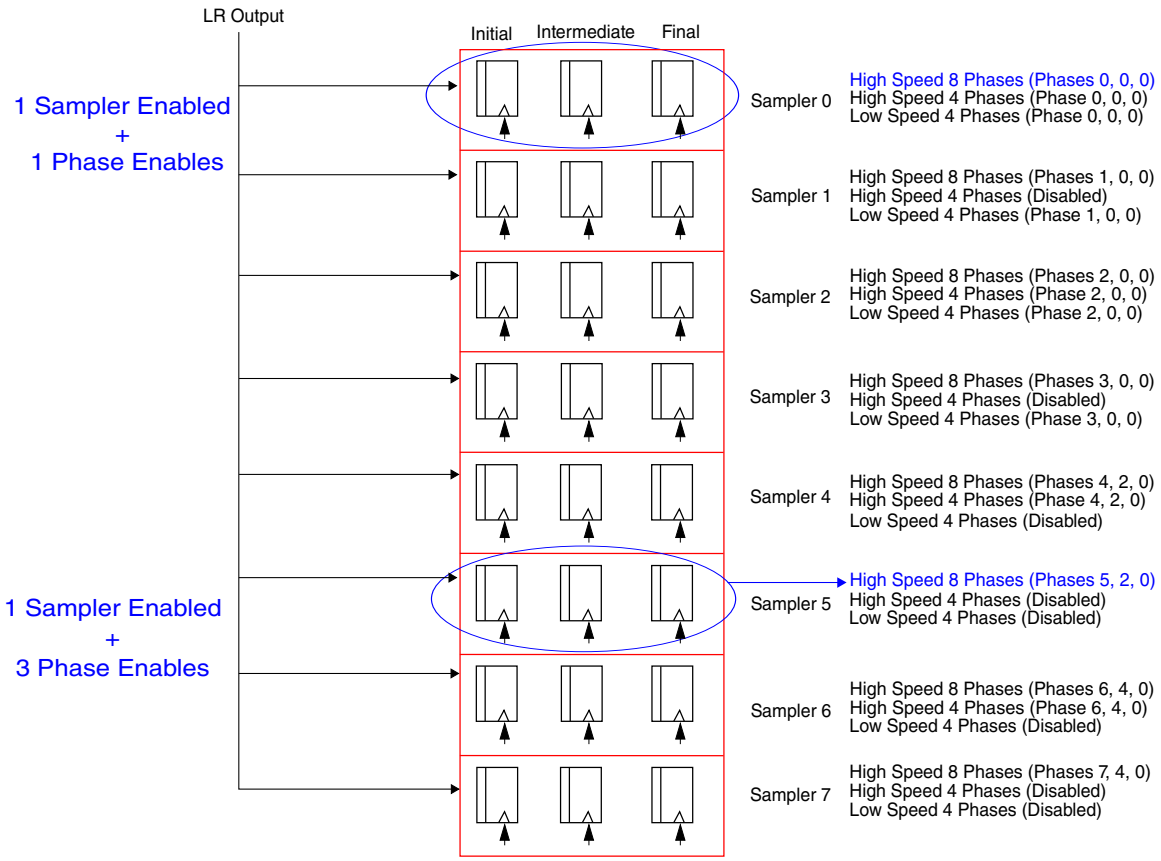


Figure 56-10. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

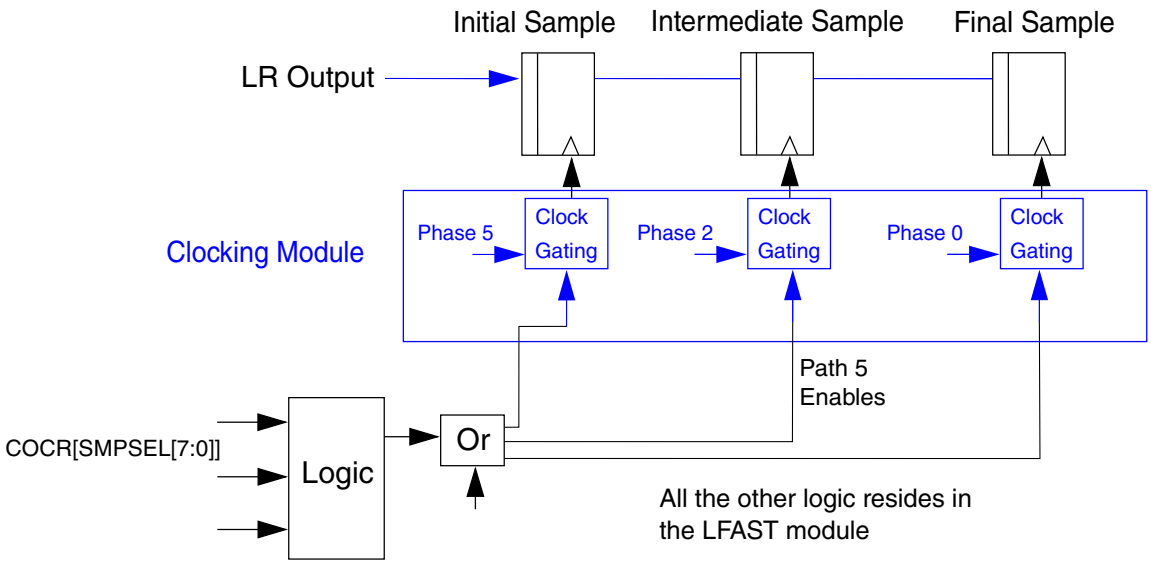


Figure 56-11. Sampler 5 Logic

56.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 56-12.

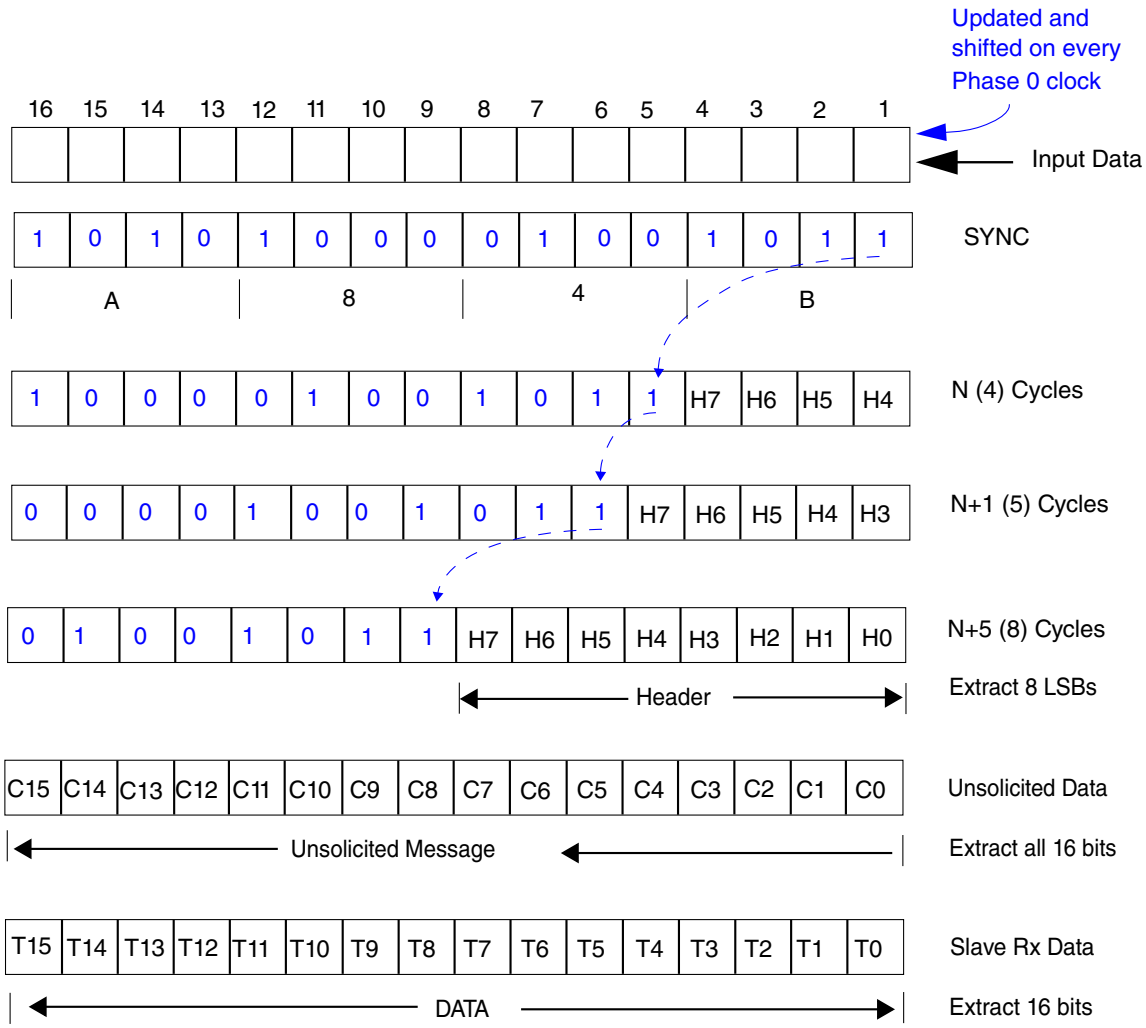


Figure 56-12. Extracting Header and Payload from the 16-bit output from Auto correlation.

56.7.3 Transmit Controller

This section describes the Transmit Controller.

56.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

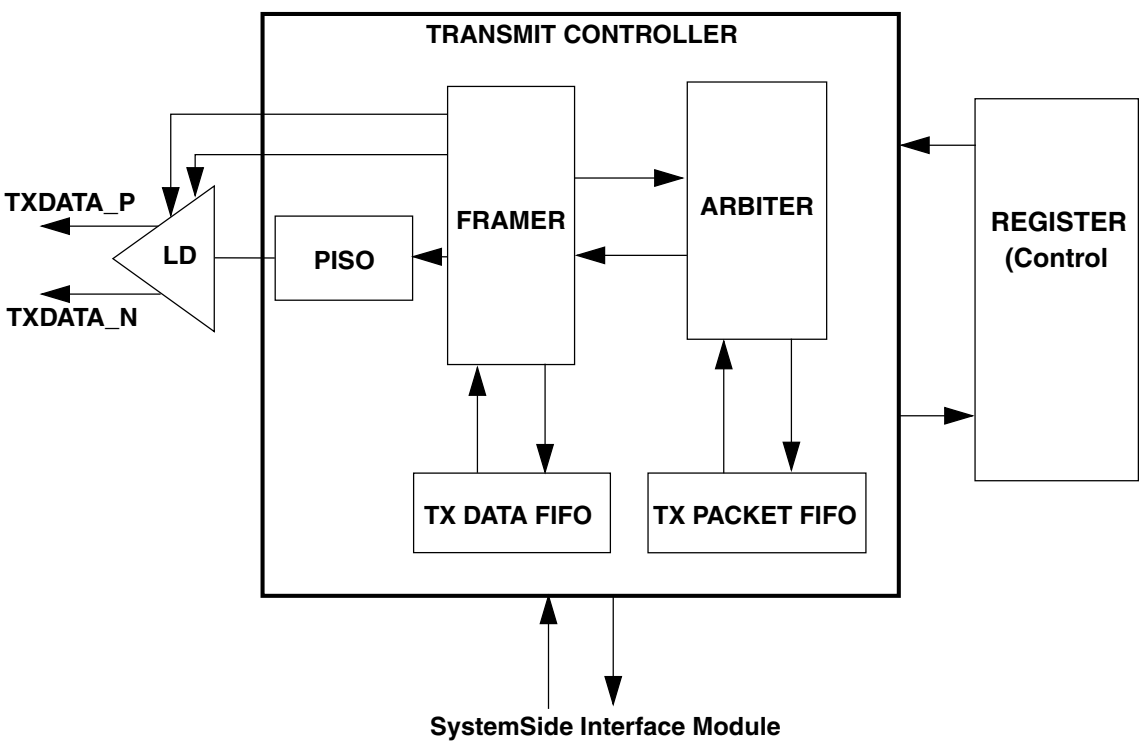


Figure 56-13. Transmit Controller Connections

56.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in Table 56-5

Table 56-5. Priority Levels for the Transmit Controller

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable	1	1	• LFAST interface enable is asserted • LFAST interface enable is negated
Tx Interface disabled	2	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	• TMCR[LPON] = 1Dig • RF Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	-	• LFAST Master: ICR[SNDICLC] = 1 • LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	-	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1

Table continues on the next page...

Table 56-5. Priority Levels for the Transmit Controller (continued)

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Unsolicited frame request	6	6	• Last Frame with CTS = 1 received from the peer LFAST device • UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	• Tx FIFO contains one or more frames • Last Frame with CTS = 1 received from the peer LFAST device • MCR[DATAEN] = 1
CTSFrame	8	8	• MCR[CTSEN] = 1 • Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

56.7.3.3 Line Driver digital connections

56.7.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

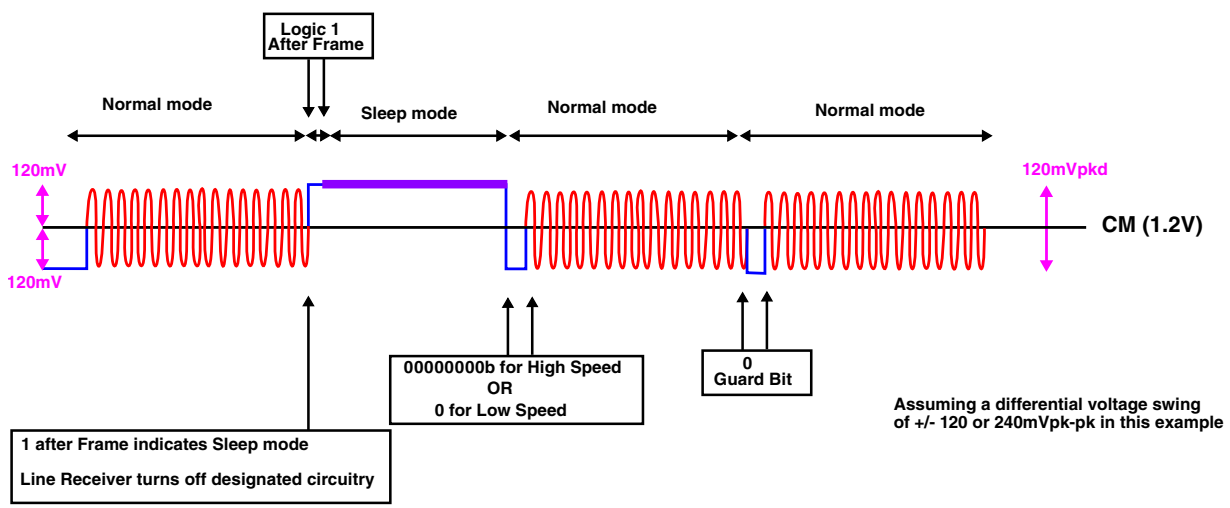


Figure 56-14. Example of Sleep mode

As shown in Figure 56-14 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated,

and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 56-6. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	• Programing LVDS[SWOFFLD] and LVDS[SWONLD] • Programing MCR[TXEN] and MCR[DRFEN]	• LFAST interface enable negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	• No pending frame for transmission • Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

56.7.4 CTS mode support

LFAST module supports flow control methods. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device doesn't want the peer device to send data. The CTS bit of all frame are sent 0 untill the Rx FIFO pointer reaches the lower threshold, as described below.

- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 untill the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed untill a frame with CTS bit 1 is received.

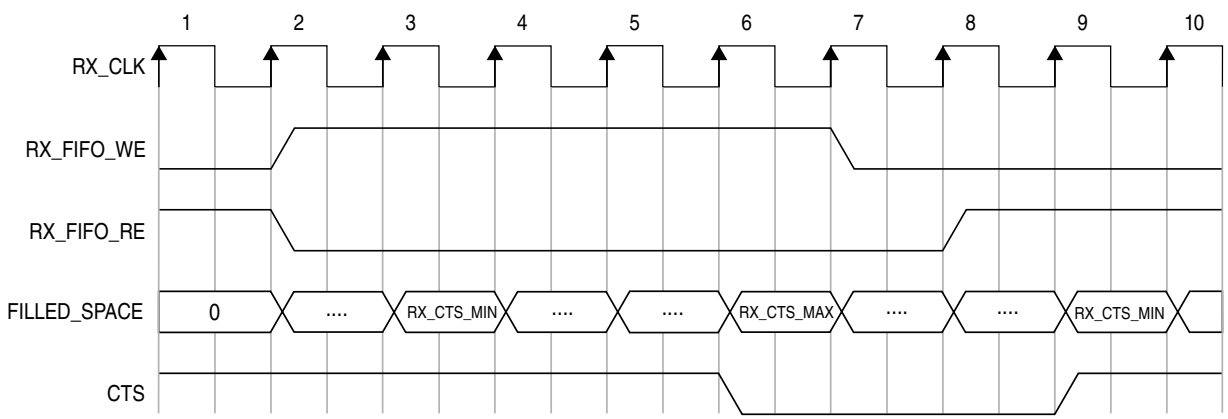


Figure 56-15. CTS generation

56.7.5 Frames supported

Table 56-7 provides the frames supported and their permissible payload sizes

Table 56-7. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in Table 56-7 then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

56.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

56.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

56.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

56.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

56.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

56.7.6.2 Unsolicited flow

56.7.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

56.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

56.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.

3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

56.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

56.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see Table 56-8 for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 56-8. Supported ICLC Payloads

ICLC Code(hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLLIn preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional)
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the Table 56-8.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note

ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 56-8.

56.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in Table 56-8 will cause RISR[RXICF] = 1.

56.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

56.7.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

56.7.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

56.7.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

56.7.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

56.7.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

56.7.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable.

56.7.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

56.7.6.4 CTS flow

56.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

56.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

56.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

56.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 56-9. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.

- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

56.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

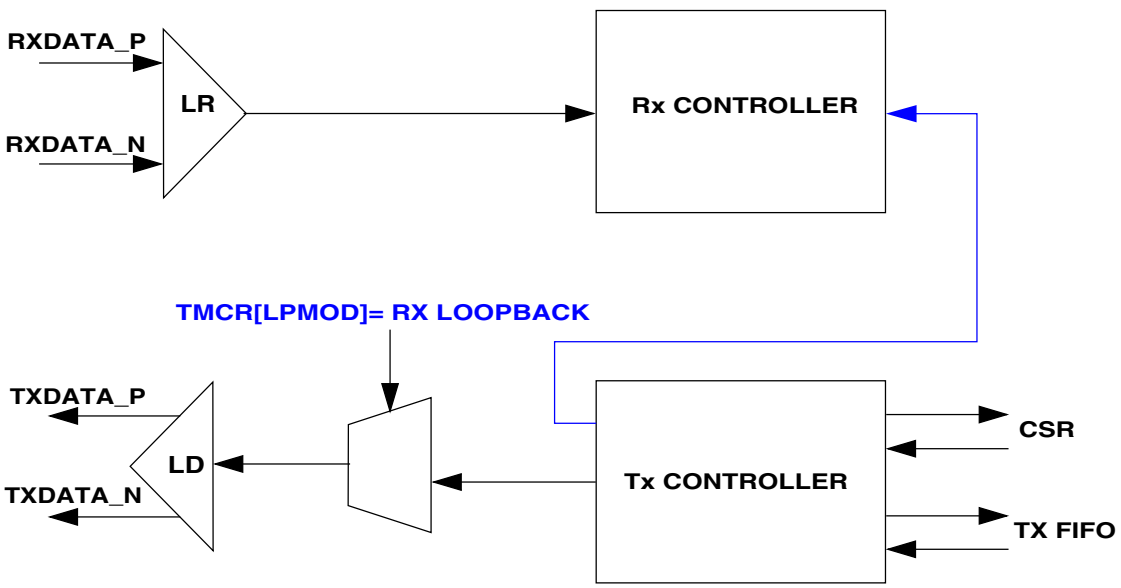


Figure 56-16. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

56.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

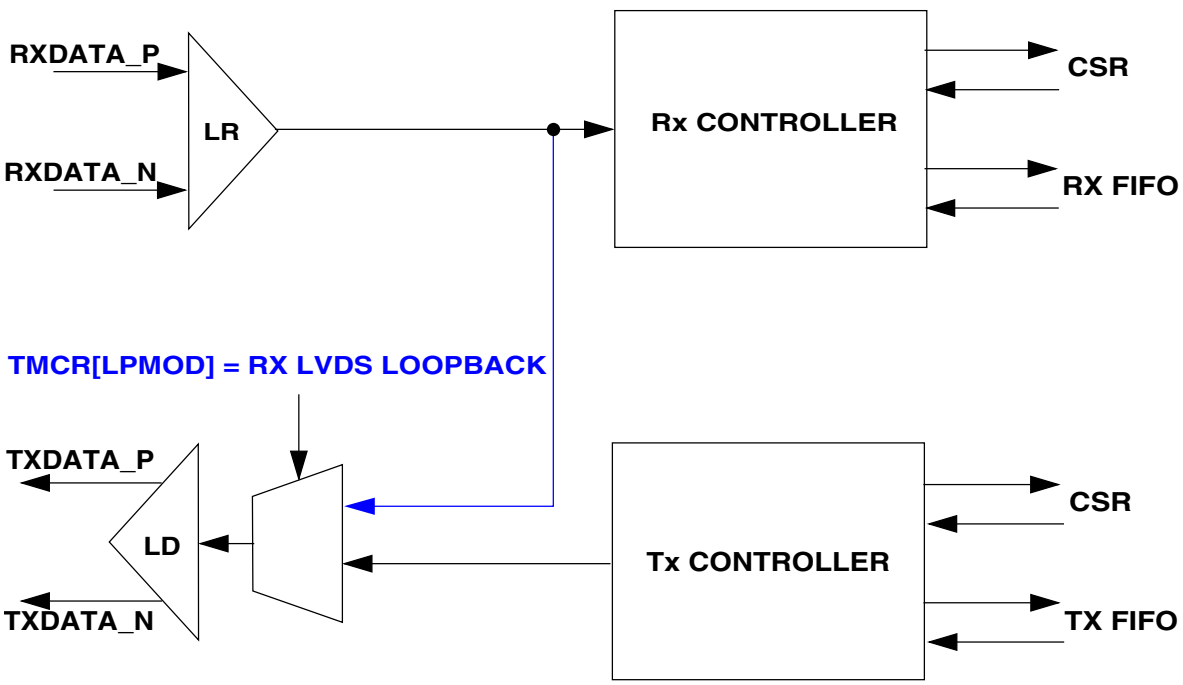


Figure 56-17. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:

- S/W writes TMCR[LPON] = 1.
- For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

56.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

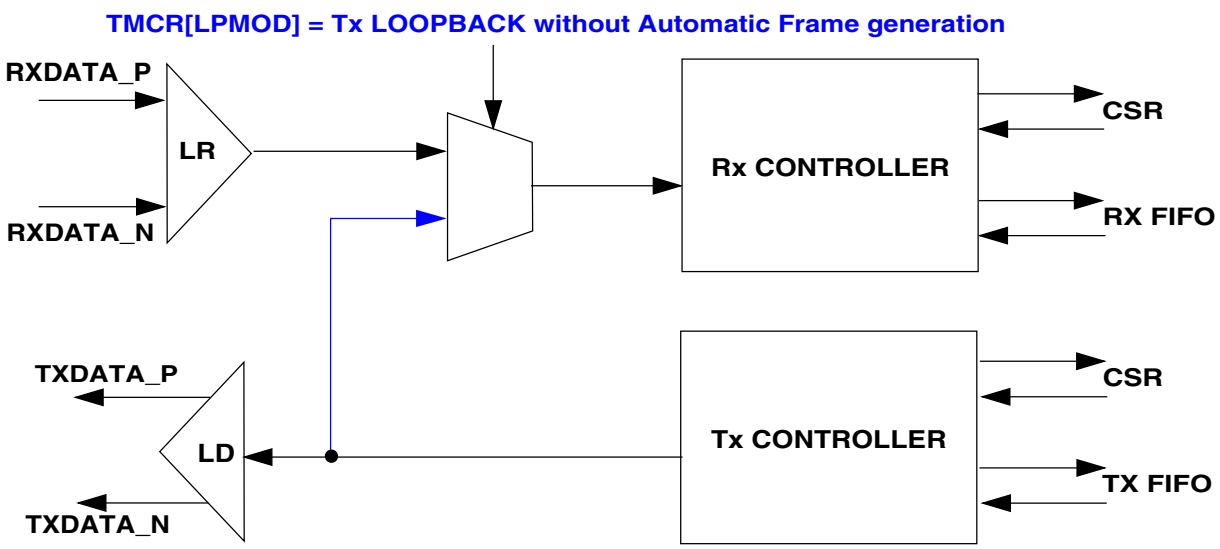


Figure 56-18. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

56.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

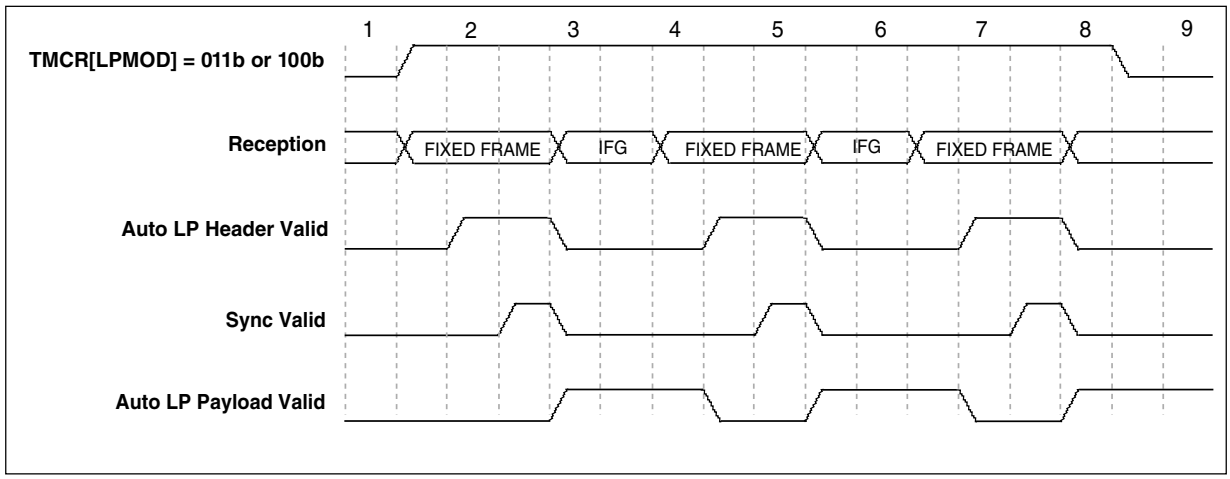


Figure 56-19. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

56.7.7.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

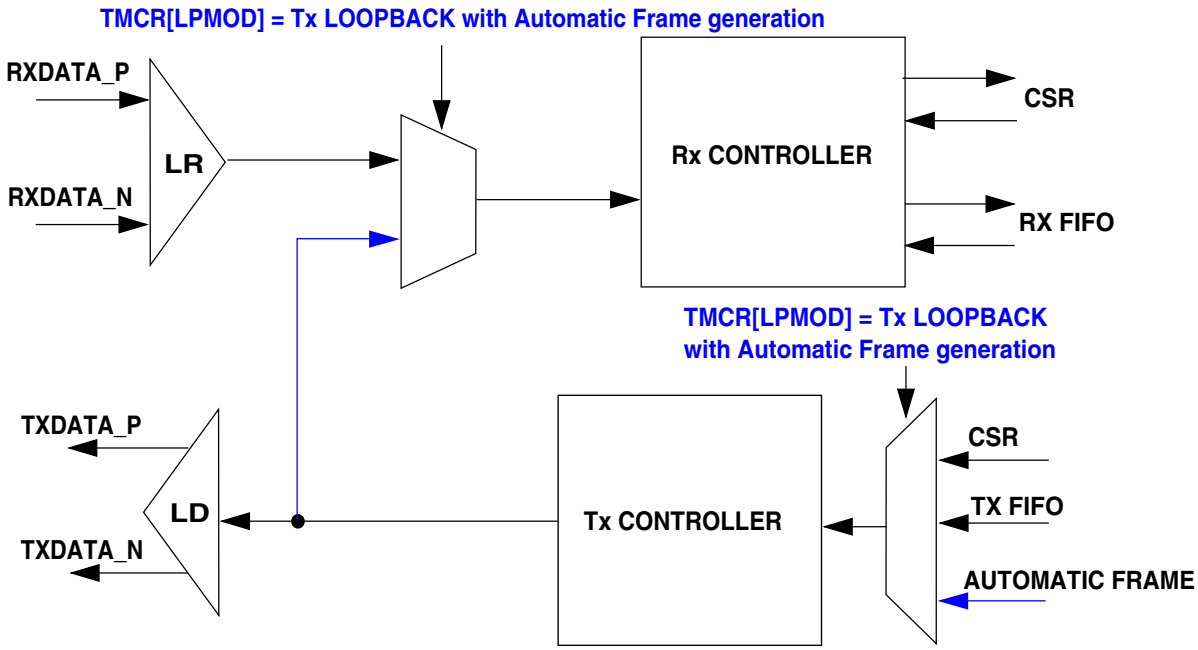


Figure 56-20. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

56.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

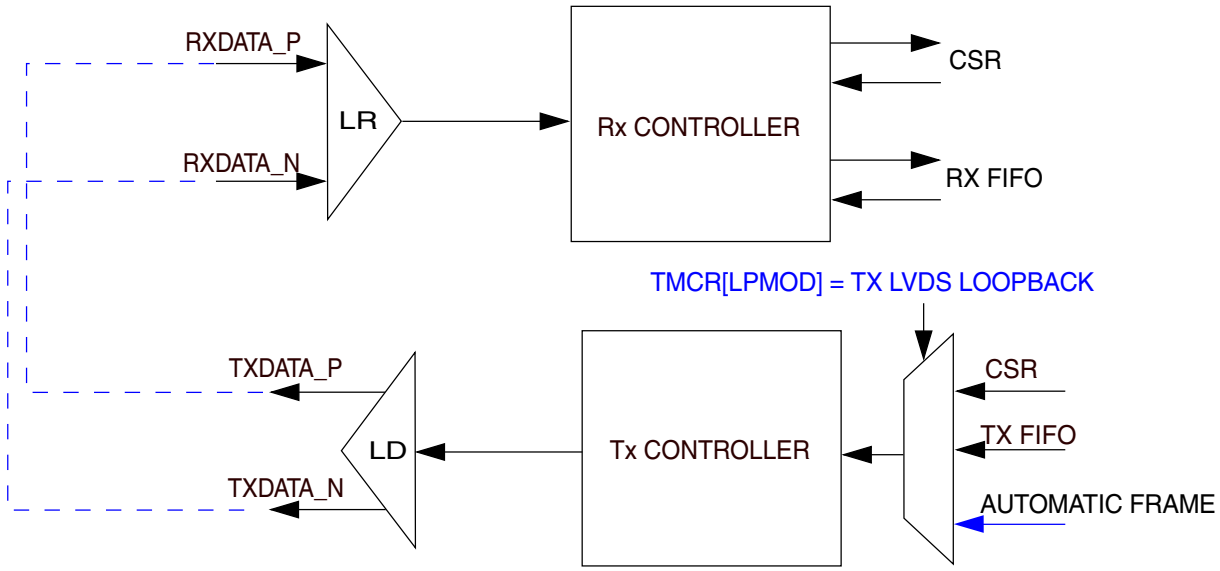


Figure 56-21. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

56.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

56.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC recieve interrupt

Each interrupt is asserted when any one of their status conditions is met.

56.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer compeletion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

56.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 56-10. Recommended transmit exception handling

Exception	Recommendation
TXOVF	• MCR[TXEN] = 0 • The system level interfaces transmit is reseted and enabled again • For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set • For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	• The MCR[TXEN] should be set

56.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

56.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 56-11. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	• The MCR[RXEN] may be cleared • The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

56.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

56.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 56-12. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet[1]	1 × 8 max 1 packet[2]	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1
2. Only for ping response data when MCR[MSEN] = 1

56.9 Resets

The various blocks in LFAST are reset as described in the table below.

Table 56-13. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]

56.10 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

56.10.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

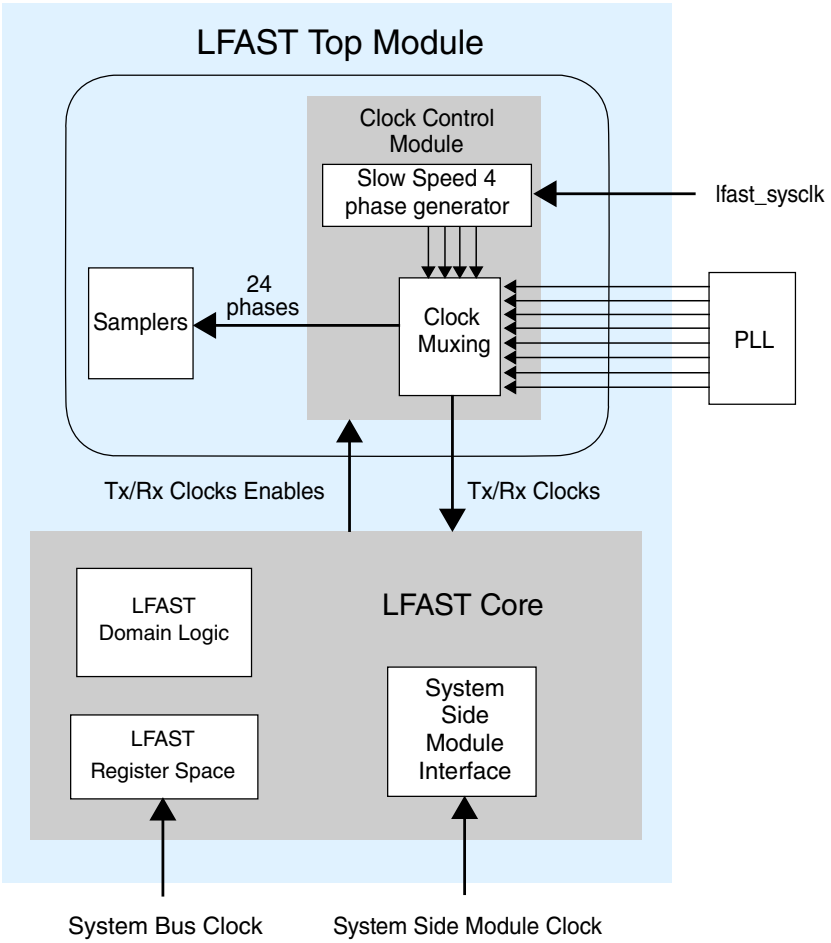


Figure 56-22. LFAST module clock domains

56.10.2 Slow speed clock

56.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 56-23, Figure 56-24 and Figure 56-25.

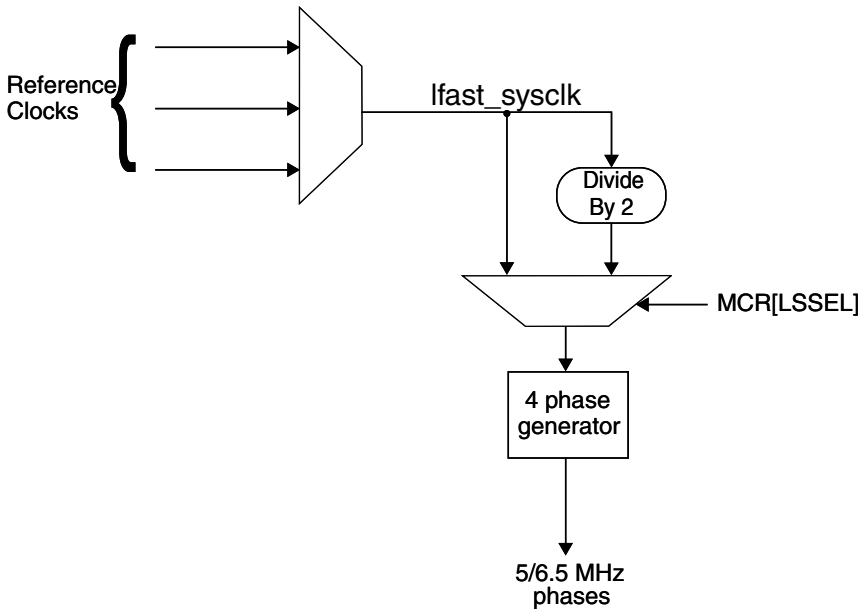


Figure 56-23. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 56-23.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 56-24 (selected clock path shown in green).

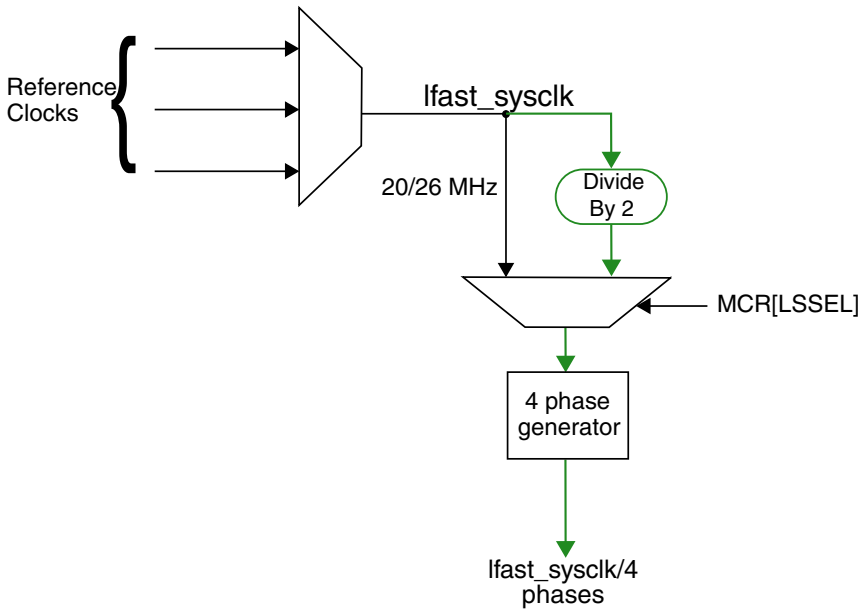


Figure 56-24. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 56-25 (selected clock path selected shown in magenta).

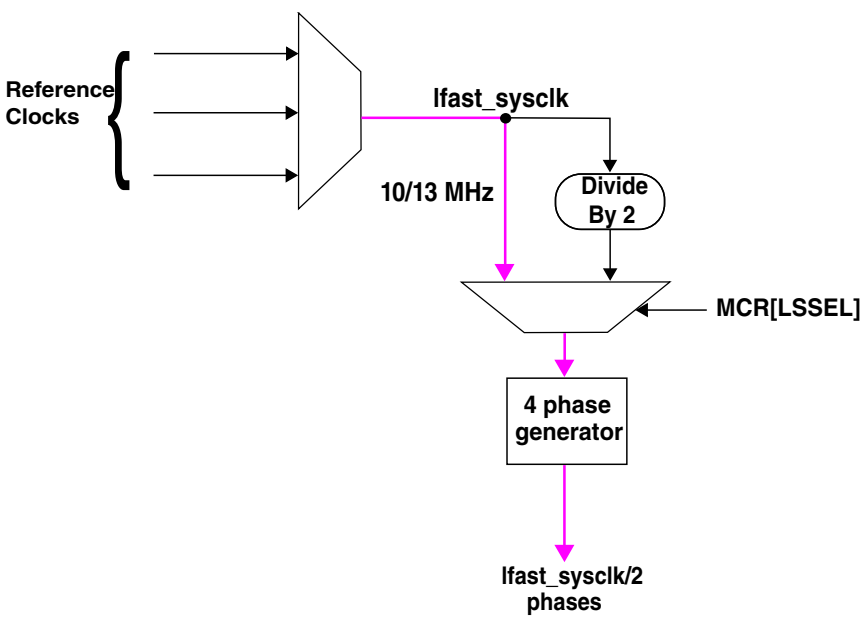


Figure 56-25. External clock muxing of lfast_sysclk of 10 MHz

56.10.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

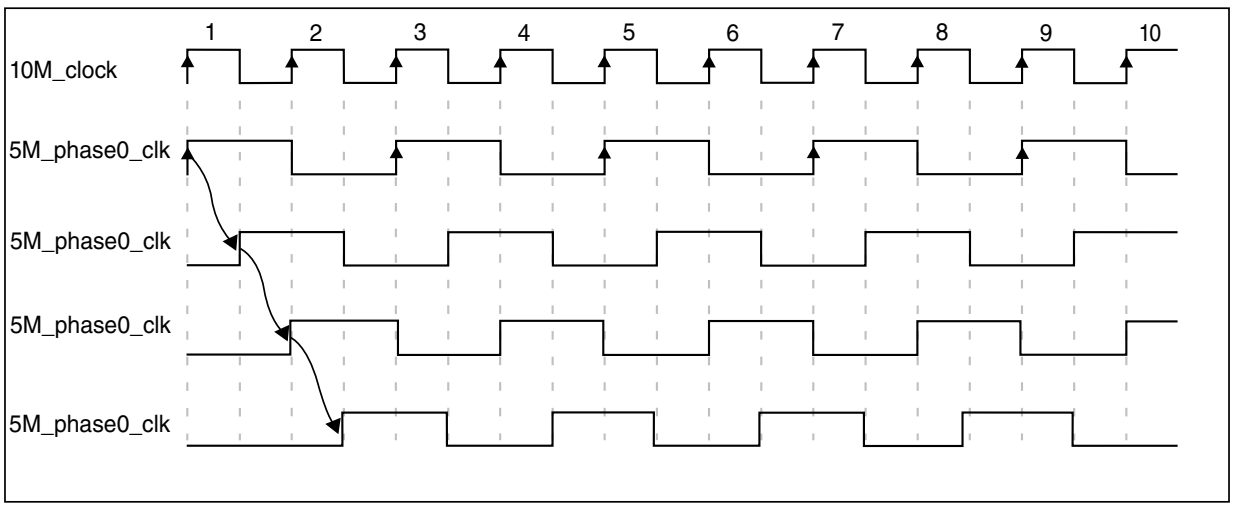


Figure 56-26. Slow speed 4 phases generation

56.10.3 Rx Controller Clocks

56.10.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

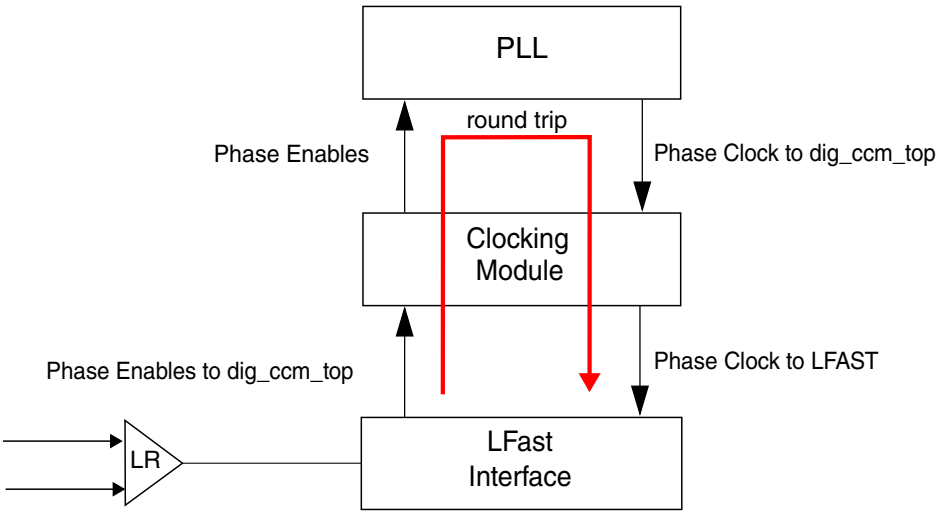


Figure 56-27. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

56.10.5 Clock module requirements for low speed phases

56.10.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 56-14. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

56.10.6 Tx Controller Clocks

56.10.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 56-15. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

56.10.6.2 Transmit Clock Muxing

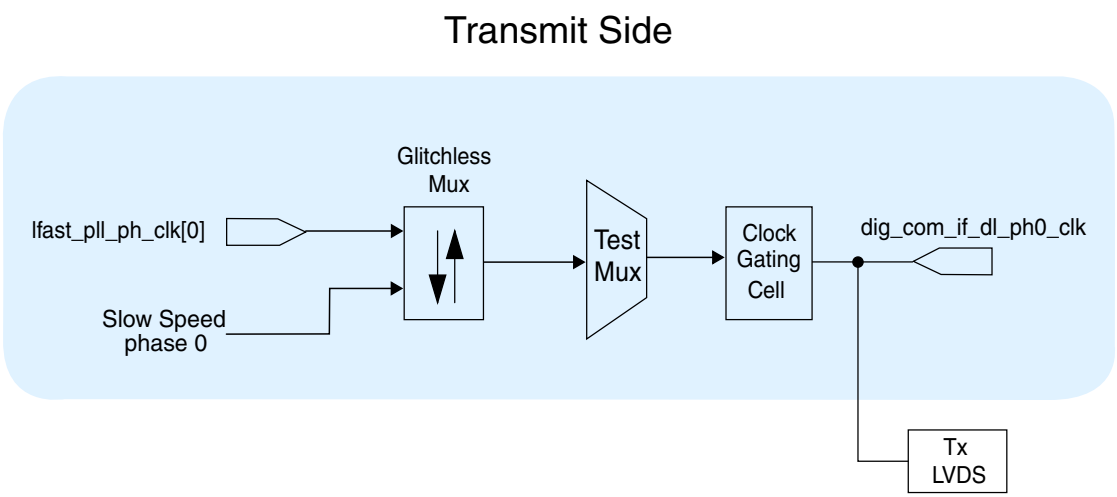


Figure 56-28. Transmit Clocks Muxing

56.10.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

56.11 PLL configuration example

This is a typical configuration for 320 MHZ LFAST high-speed clock when the reference clock is 20 MHz.

1. Program the LFAST_PLLCR[FBDIV] for the Feedback Division factor, in this case 15 (divide by 16)
2. Program the LFAST_PLLCR[PREDIV] for the Division factor for PLL Reference Clock, in this case 0 (divide by 1)
3. Program the LFAST_PLLCR[SWPON] to 1, for switching the PLL ON

The user should wait for:

1. The LFAST_PLLLSR[PLLDIS] =’0’, indicating that the PLL is enabled
2. Then, the LFAST_PLLLSR[PLDCR] =’1’, indicating that the PLL is locked and ready for use

Chapter 57
Serial Peripheral Interface (SPI)

57.1 Chip-specific Serial Peripheral Interface (SPI) information

57.1.1 SPI modules configuration

The device contains two SPI module.

57.1.2 Number of CTARs

SPI CTAR registers define different transfer attribute configurations. The SPI module supports up to eight CTAR registers.

In master mode, the CTAR registers define combinations of transfer attributes, such as frame size, clock phase, clock polarity, data bit ordering, baud rate, and various delays. In slave mode only CTAR0 is used, and a subset of its bitfields sets the slave transfer attributes; while in master mode, two CTARs provides enough flexibility to set different attributes to the type of device connected on the SPI.

57.1.3 TX FIFO size

Table 57-1. SPI transmit FIFO size

SPI Module	Transmit FIFO size
SPI_1	5
SPI_2	5

57.1.4 RX FIFO Size

Table 57-2. SPI receive FIFO size

SPI Module	Receive FIFO size
SPI_1	5
SPI_2	5

57.1.5 Number of PCS signals

The following table shows the number of peripheral chip select signals available per SPI module.

Table 57-3. SPI PCS signals

SPI Module	PCS Signals
SPI_1	SPI_PCS[7:0]
SPI_2	SPI_PCS[3:0]

NOTE

When configured in slave mode,once SPI is enabled do NOT assert SPI module disable (SPI_MCR[MDIS]). Allow the Mode Entry function to disable the SPI module.

57.2 Introduction

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

57.2.1 Block Diagram

The block diagram of this module is as follows:

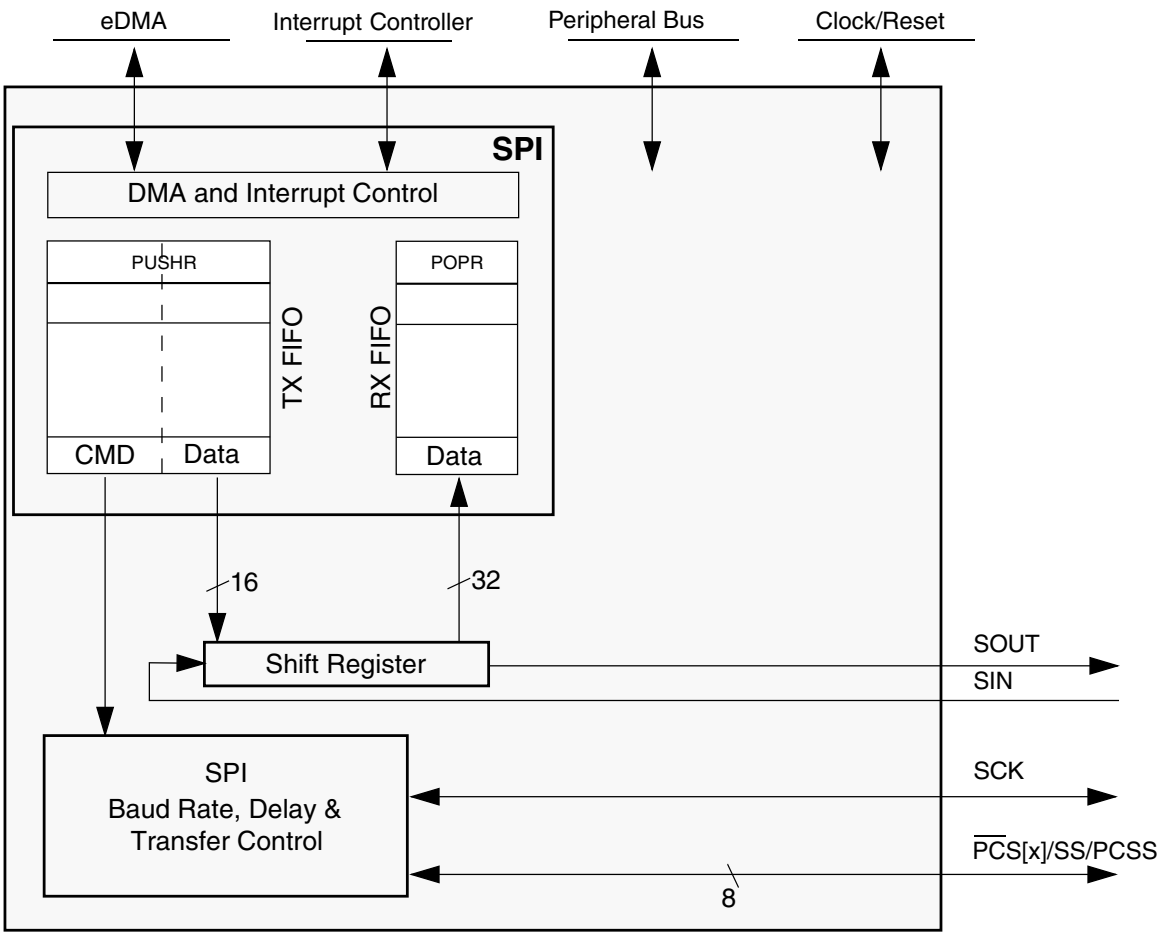


Figure 57-1. SPI Block Diagram

57.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 5 entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 5 entries
- Asynchronous clocking scheme for Register and Protocol Interfaces

- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - eight transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size: 4 to 16
    - SPI frames longer than 16 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
  - Parity control
- 8 peripheral chip selects (PCSes), expandable to 256 with external demultiplexer
- Deglitching support for up to 128 peripheral chip selects (PCSes) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - Transfer of current frame complete (TCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
- Global interrupt request line
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:

- Support for Stop mode

57.2.3 Interface configurations

57.2.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

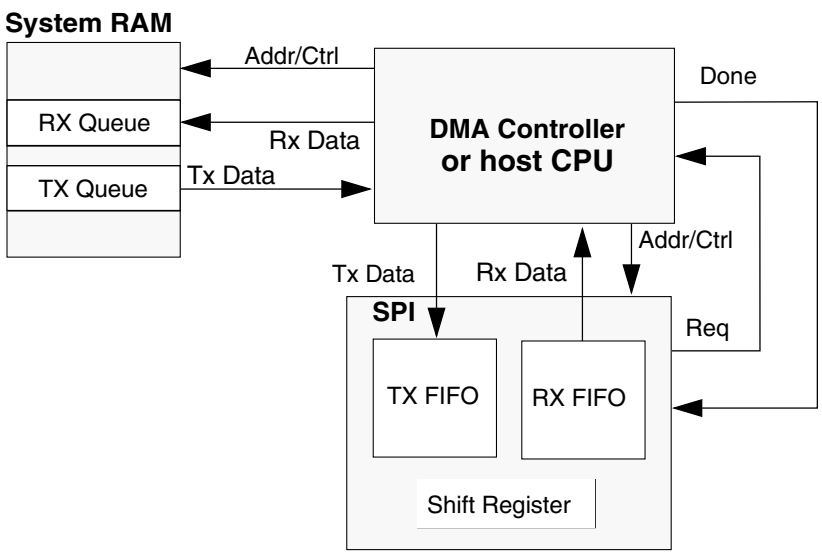


Figure 57-2. SPI with queues and DMA

57.2.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:

- Master mode
- Slave mode
- Module Disable mode

- Chip-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

57.2.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

57.2.4.2 Slave Mode

Slave mode allows the module to communicate with SPI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{\text{SS}}$ signals are configured as inputs and driven by an SPI bus master.

57.2.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

57.2.4.4 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

57.2.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

57.3 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 57-4. Module signal descriptions

Signal	Master mode	Slave mode	I/O
PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4	Peripheral Chip Select 4	(Unused)	O
PCS5/ $\overline{PCSS}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
PCS[6:7]	Peripheral Chip Selects 6–7	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

57.3.1 PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

NOTE

Do not tie the SPI slave select pin to ground. Otherwise, SPI cannot function properly.

57.3.2 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

57.3.3 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Unused

57.3.4 PCS5/$\overline{\text{PCSS}}$—Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strobes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

57.3.5 PCS6–PCS7—Peripheral Chip Selects 6–7

Master mode: Peripheral Chip Selects 6–7 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

57.3.6 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

57.3.7 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

57.3.8 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

57.4 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

SPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (SPI_MCR)	32	R/W	0000_4001h	57.4.1/2683
8	Transfer Count Register (SPI_TCR)	32	R/W	0000_0000h	57.4.2/2686
C	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR0)	32	R/W	7800_0000h	57.4.3/2687
C	Clock and Transfer Attributes Register (In Slave Mode) (SPI_CTAR0_SLAVE)	32	R/W	7800_0000h	57.4.4/2692
10	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR1)	32	R/W	7800_0000h	57.4.3/2687
14	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR2)	32	R/W	7800_0000h	57.4.3/2687
18	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR3)	32	R/W	7800_0000h	57.4.3/2687
1C	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR4)	32	R/W	7800_0000h	57.4.3/2687

Table continues on the next page...

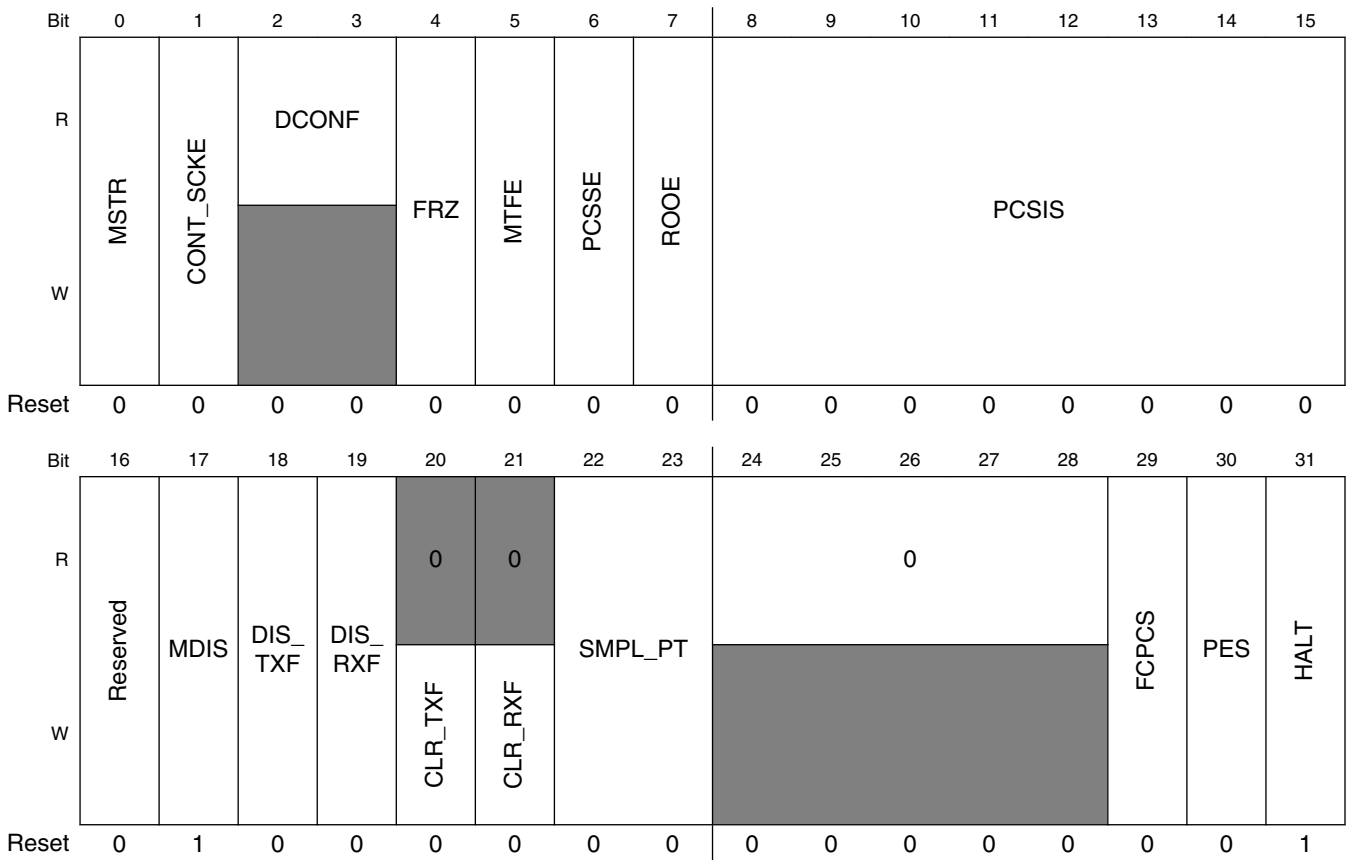
SPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR5)	32	R/W	7800_0000h	57.4.3/2687
24	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR6)	32	R/W	7800_0000h	57.4.3/2687
28	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR7)	32	R/W	7800_0000h	57.4.3/2687
2C	Status Register (SPI_SR)	32	R/W	0201_0000h	57.4.5/2694
30	DMA/Interrupt Request Select and Enable Register (SPI_RSER)	32	R/W	0000_0000h	57.4.6/2697
34	PUSH TX FIFO Register In Master Mode (SPI_PUSHR)	32	R/W	0000_0000h	57.4.7/2699
34	PUSH TX FIFO Register In Slave Mode (SPI_PUSHR_SLAVE)	32	R/W	0000_0000h	57.4.8/2702
38	POP RX FIFO Register (SPI_POPR)	32	R	0000_0000h	57.4.9/2702
3C	Transmit FIFO Registers (SPI_TXFR0)	32	R	0000_0000h	57.4.10/ 2703
40	Transmit FIFO Registers (SPI_TXFR1)	32	R	0000_0000h	57.4.10/ 2703
44	Transmit FIFO Registers (SPI_TXFR2)	32	R	0000_0000h	57.4.10/ 2703
48	Transmit FIFO Registers (SPI_TXFR3)	32	R	0000_0000h	57.4.10/ 2703
4C	Transmit FIFO Registers (SPI_TXFR4)	32	R	0000_0000h	57.4.10/ 2703
7C	Receive FIFO Registers (SPI_RXFR0)	32	R	0000_0000h	57.4.11/ 2703
80	Receive FIFO Registers (SPI_RXFR1)	32	R	0000_0000h	57.4.11/ 2703
84	Receive FIFO Registers (SPI_RXFR2)	32	R	0000_0000h	57.4.11/ 2703
88	Receive FIFO Registers (SPI_RXFR3)	32	R	0000_0000h	57.4.11/ 2703
8C	Receive FIFO Registers (SPI_RXFR4)	32	R	0000_0000h	57.4.11/ 2703

57.4.1 Module Configuration Register (SPI_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 0h base + 0h offset = 0h



SPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously.

Table continues on the next page...

SPI_MCR field descriptions (continued)

Field	Description
	0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	SPI Configuration. Selects among the different configurations of the module. 00 SPI 01 Reserved 10 Reserved 11 Reserved
4 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
5 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used. **NOTE:** When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when the module is in Master Mode. This field has no effect when the module is in Slave Mode. The Slave Select input to the module in slave mode is always Active Low. **NOTE:** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the SPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.

Table continues on the next page...

SPI_MCR field descriptions (continued)

Field	Description
16 Reserved	This field is reserved.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
21 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. **NOTE:** After every RX FIFO clear operation (MCR [CLR_RXF] = 0b1) following a RX FIFO overflow (SR [RFOF] = 0b1) scenario, immediately perform a single POP from the RX FIFO and discard the read data. The POP and discard operation should be completed before the reception of new incoming frame. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved

Table continues on the next page...

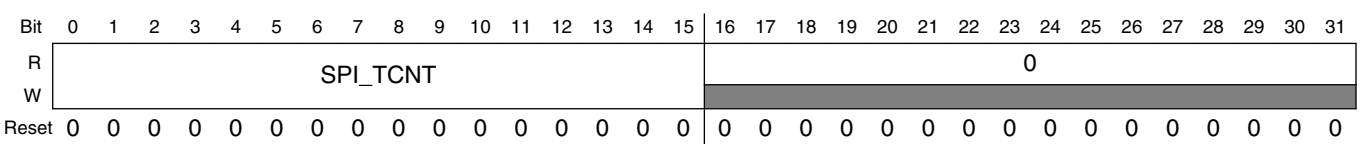
SPI_MCR field descriptions (continued)

Field	Description
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHR register. The firmware should select appropriate masks when providing continuous frames via the PUSHR register. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR register.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0 Start transfers. 1 Stop transfers.

57.4.2 Transfer Count Register (SPI_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 0h base + 8h offset = 8h



SPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI

Table continues on the next page...

SPI_TCR field descriptions (continued)

Field	Description
	command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.3 Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 are used to set the slave transfer attributes.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, it uses the CTAR0 register.

Address: 0h base + Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate. **Table 57-5. SPI SCK Duty Cycle** DBR / CPHA / PBR / SCK Duty Cycle 0 / any / any / 50/50 1 / 0 / 00 / 50/50 1 / 0 / 01 / 33/66 1 / 0 / 10 / 40/60 1 / 0 / 11 / 43/57 1 / 1 / 00 / 50/50 1 / 1 / 01 / 66/33 1 / 1 / 10 / 60/40 1 / 1 / 11 / 57/43 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. There is a constraint on the minimum allowable Frame size, which depends on the ratio of the Register Read/Write clock to the Protocol clock. The minimum Frame size (FMSZ + 1) is determined by the following equation. Upper Ceiling must be applied for non-integer values. Minimum Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol clock frequency f_r = Register interface clock frequency n = (PBR x BR) / (1 + DBR) = Multiple of protocol clock required to get Baud (SCK) Clock However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Also note that `fp' can be equal to, less than or greater than `fr' purely based on user requirement.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10–11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay (t_{ASC}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
14–15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3.

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16–19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values. **Table 57-6. Delay Scaler Encoding** **Field Value** / **Delay Scaler Value** 0000 / 2 0001 / 4 0010 / 8 0011 / 16 0100 / 32 0101 / 64 0110 / 128 0111 / 256 1000 / 512 1001 / 1024 1010 / 2048 1011 / 4096 1100 / 8192 1101 / 16384 1110 / 32768 1111 / 65536 Refer PCS to SCK Delay (t_{CSC}) for more details.
20–23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
24–27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame.

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.
28–31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = $(f_P / PBR) \times [(1+DBR)/BR]$ The following table lists the baud rate scaler values.

Table 57-7. Baud Rate Scaler

CTARn[BR]	Baud Rate Scaler Value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

57.4.4 Clock and Transfer Attributes Register (In Slave Mode) (SPI_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

Address: 0h base + Ch offset + (0d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved	FMSZ				CPOL	CPHA	PE	PP	Reserved	Reserved					
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_CTAR*n*_SLAVE field descriptions

Field	Description
0 Reserved	Always write the reset value to this field. This field is reserved.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.

Table continues on the next page...

SPI_CTAR*n*_SLAVE field descriptions (continued)

Field	Description
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 Reserved	This field is reserved.
10–31 Reserved	This field is reserved.

57.4.5 Status Register (SPI_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	SPEF	0	RFOF	0	RFDF	Reserved
W	w1c			w1c	w1c		w1c				w1c		w1c		w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.

Table continues on the next page...

SPI_SR field descriptions (continued)

Field	Description
1 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for SPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request. **NOTE:** The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0 TX FIFO is full. 1 TX FIFO is not full.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.

Table continues on the next page...

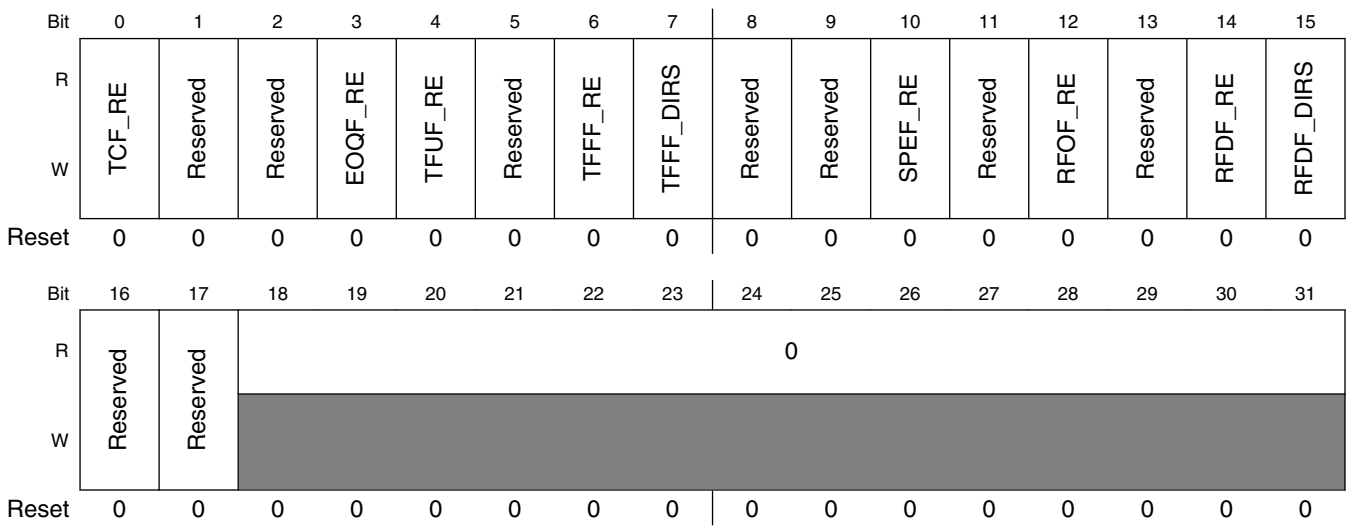
SPI_SR field descriptions (continued)

Field	Description
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDF	Receive FIFO Drain Flag Indicates whether there is an available location to be drained from the FIFO. Either a DMA request or an interrupt indication can be used to read from the FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
15 Reserved	This field is reserved.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

57.4.6 DMA/Interrupt Request Select and Enable Register (SPI_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 0h base + 30h offset = 30h



SPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 Reserved	Always write the reset value to this field. This field is reserved.
2 Reserved	Always write the reset value to this field. This field is reserved.
3 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request.

Table continues on the next page...

SPI_RSER field descriptions (continued)

Field	Description
	0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5 Reserved	Always write the reset value to this field. This field is reserved.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 Reserved	Always write the reset value to this field. This field is reserved.
9 Reserved	Always write the reset value to this field. This field is reserved.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11 Reserved	Always write the reset value to this field. This field is reserved.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 Reserved	Always write the reset value to this field. This field is reserved.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.

Table continues on the next page...

SPI_RSER field descriptions (continued)

Field	Description
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 Reserved	Always write the reset value to this field. This field is reserved.
17 Reserved	Always write the reset value to this field. This field is reserved.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

57.4.7 PUSH TX FIFO Register In Master Mode (SPI_PUSHR)

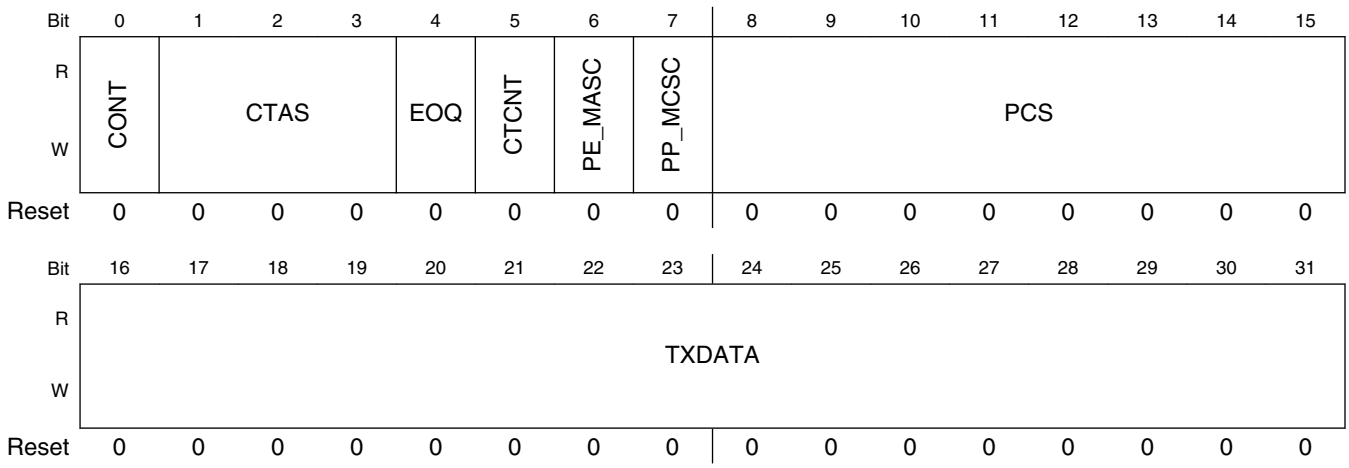
Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

The TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. Because both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: 0h base + 34h offset = 34h



SPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 CTAR2 011 CTAR3 100 CTAR4 101 CTAR5 110 CTAR6 111 CTAR7
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.

Table continues on the next page...

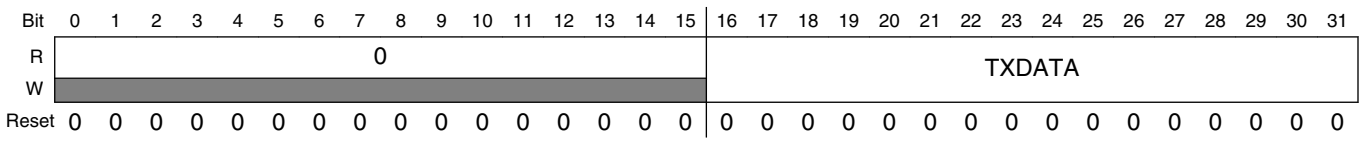
SPI_PUSHR field descriptions (continued)

Field	Description
6 PE_MASC	Parity Enable or Mask T_{ASC} delay in the current frame PE – This bit enables parity bit transmission and parity reception check for the SPI frame. MASC - The current frame has the “after SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE:** This bit is used as Mask T_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE - No parity bit included/checked. MASC - T_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - T_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask T_{CSC} delay in the next frame PP - It controls the polarity of the parity bit transmitted and checked. MCSC - The next frame has the “PCS to SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE:** This bit is used as Mask T_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - T_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1 PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - T_{CSC} delay is masked in the next frame.
8–15 PCS	PCS Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific SPI information for the number of PCS signals used in this chip. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

57.4.8 PUSH TX FIFO Register In Slave Mode (SPI_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

Address: 0h base + 34h offset = 34h



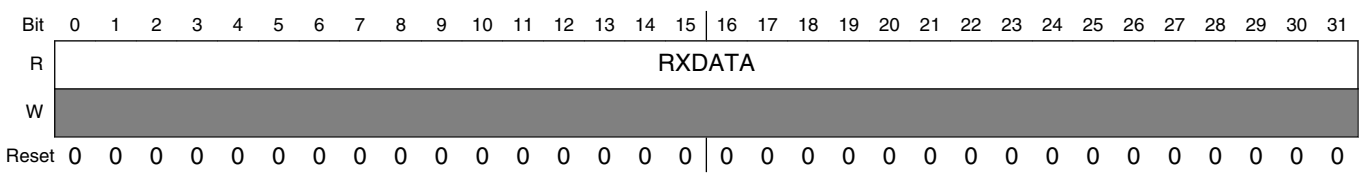
SPI_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

57.4.9 POP RX FIFO Register (SPI_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 0h base + 38h offset = 38h



SPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

57.4.10 Transmit FIFO Registers (SPI_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCMD_TXDATA																TXDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_TXFR*n* field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

57.4.11 Receive FIFO Registers (SPI_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_RXFR*n* field descriptions

Field	Description
0–31 RXDATA	Receive Data

SPI_RXFR*n* field descriptions (continued)

Field	Description
	Contains the received SPI data.

57.5 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command.

See Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*) for information on the fields of CTAR registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

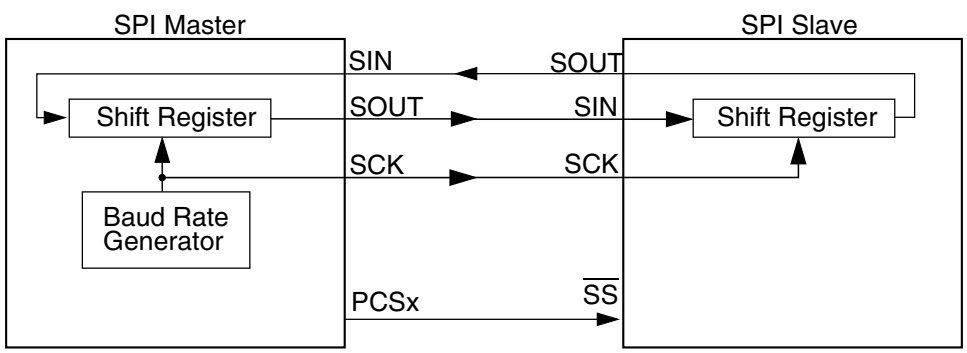


Figure 57-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 8 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip specific section for details on the number of PCS signals used in this chip.

The SPI configuration shares transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

57.5.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

57.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Command First In First Out (CMD FIFO) Buffering Mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode. In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

57.5.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (SPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.

57.5.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

57.5.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:
- SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

57.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 5 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

57.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA

controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

57.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

57.5.2.5 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. The TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.

The CMD FIFO holds 5 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

Every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

57.5.2.6 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 5 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

57.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

57.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

57.5.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

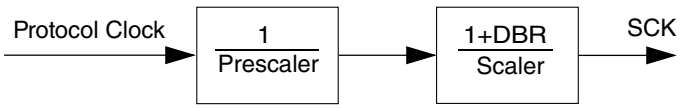


Figure 57-4. Communications clock prescalers and scalers

57.5.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 57-8. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

57.5.3.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 57-6 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 57-9. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

57.5.3.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 57-6 and Figure 57-7 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 57-10. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

57.5.3.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 57-6 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 57-11. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

57.5.3.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

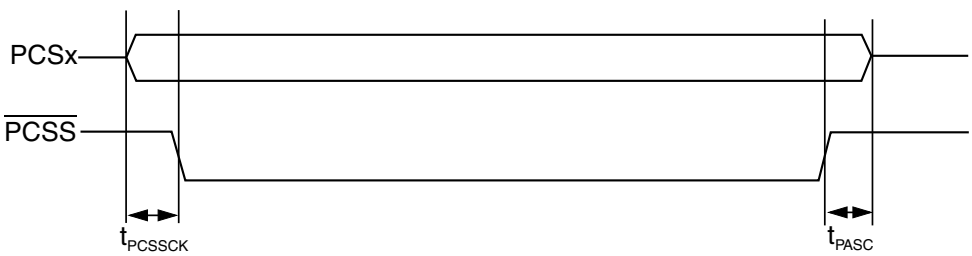


Figure 57-5. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 57-12. Peripheral Chip Select Strobe Assert computation example

f_P	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 57-13. Peripheral Chip Select Strobe Negate computation example

f_P	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The $\overline{PCSS}$ signal is not supported when Continuous Serial Communication SCK mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

57.5.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

57.5.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

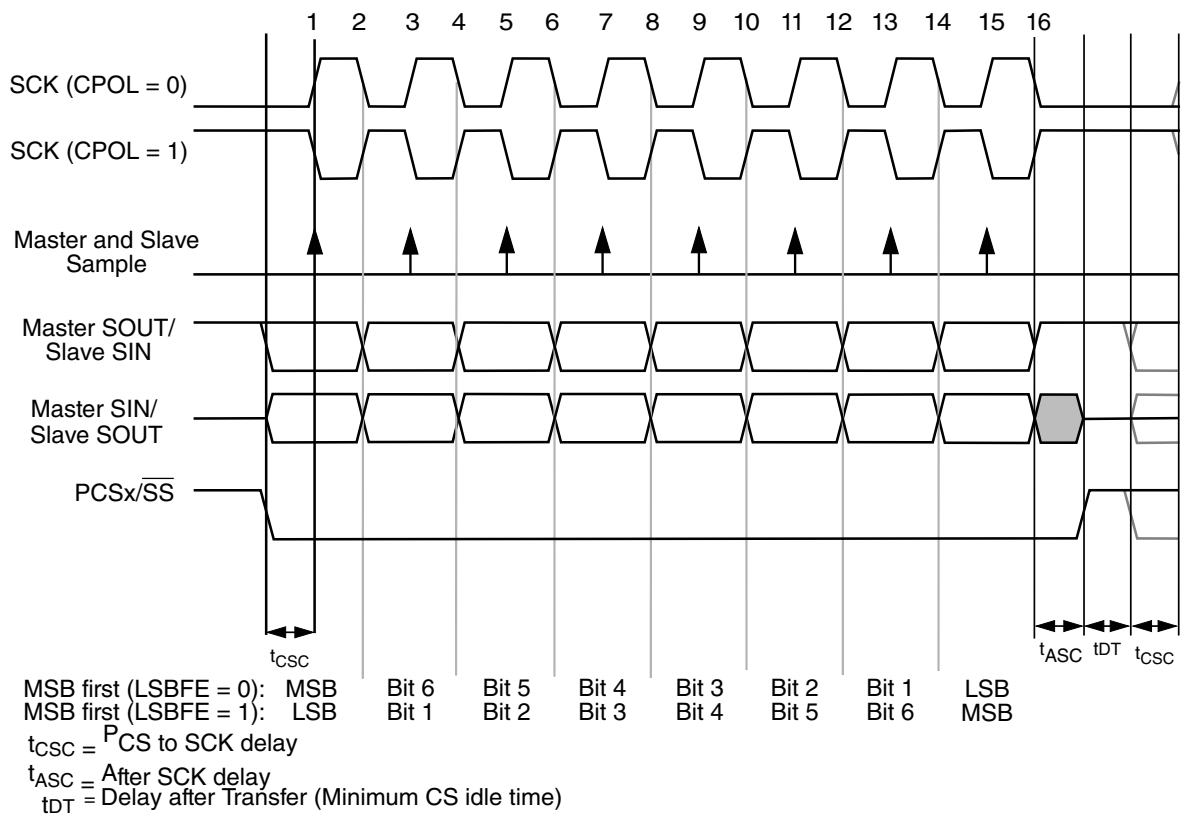


Figure 57-6. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and

slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

57.5.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

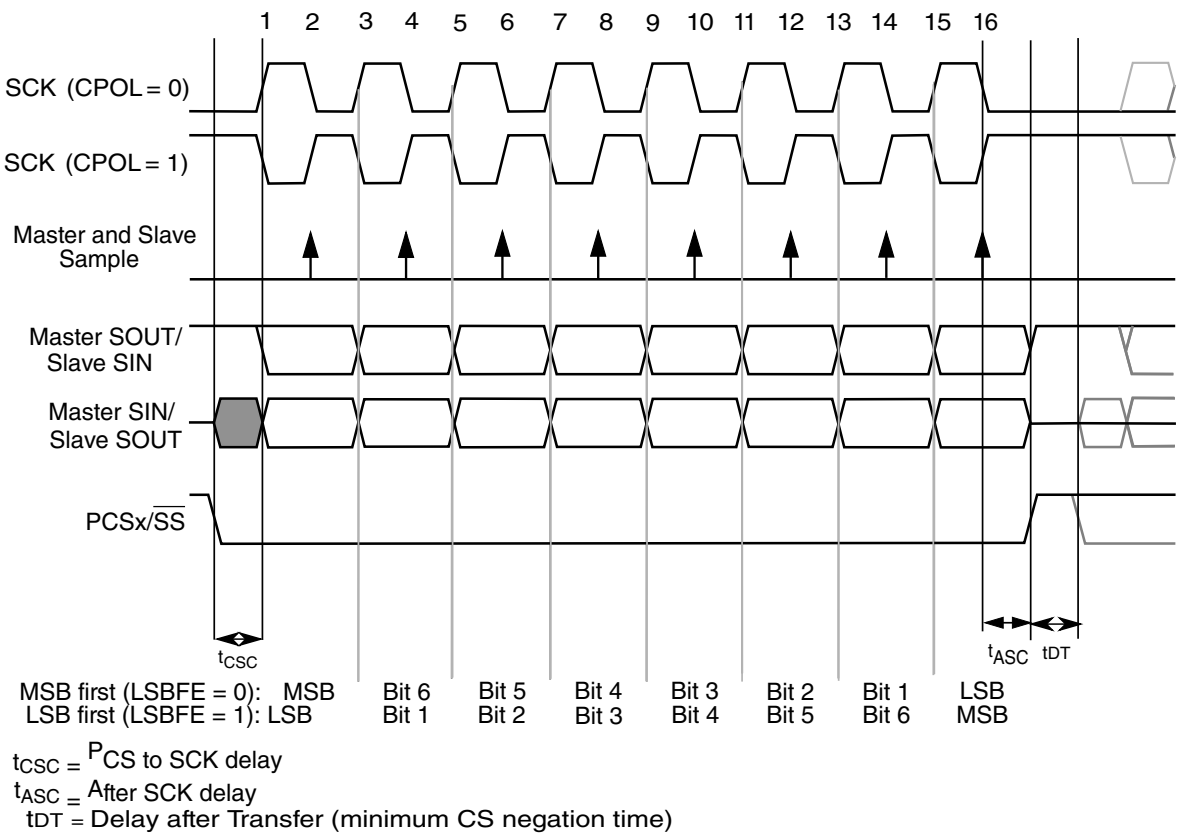


Figure 57-7. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

57.5.4.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.

- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

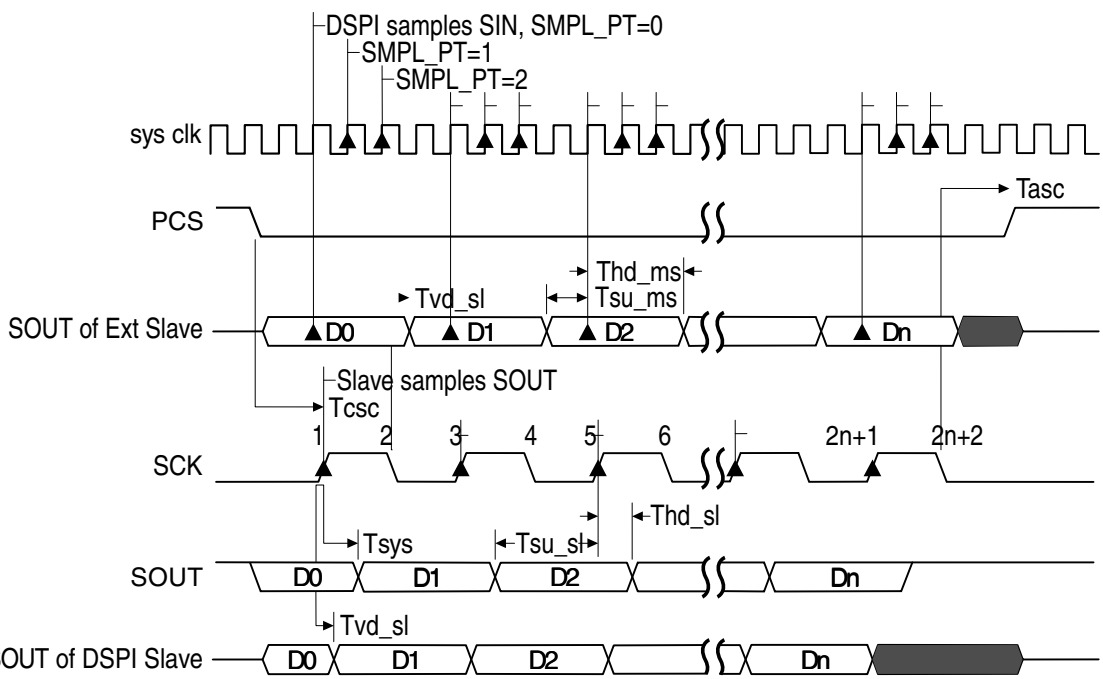


Figure 57-8. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

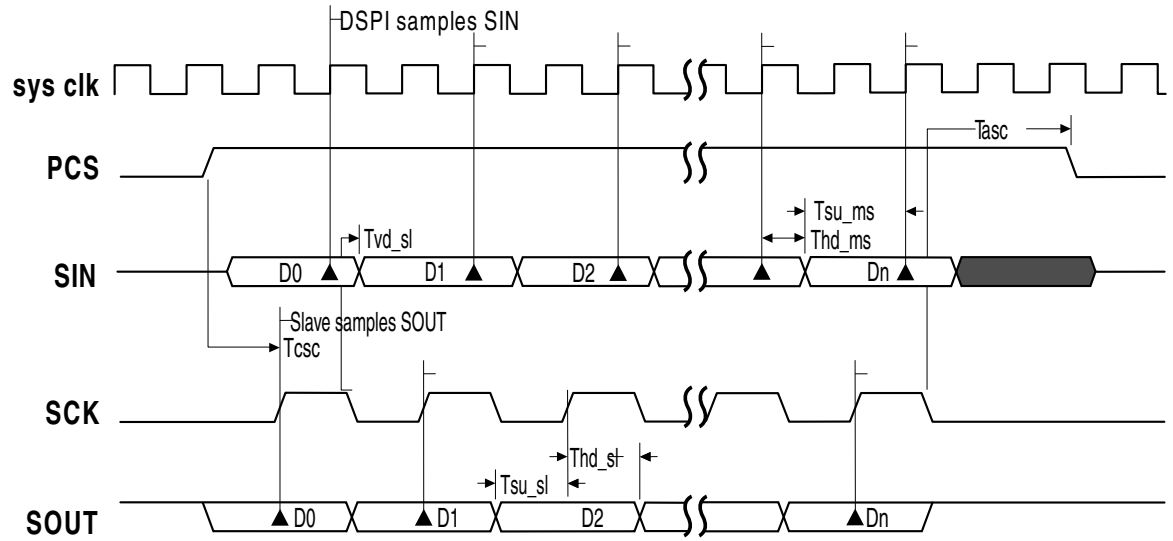


Figure 57-9. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/2$)

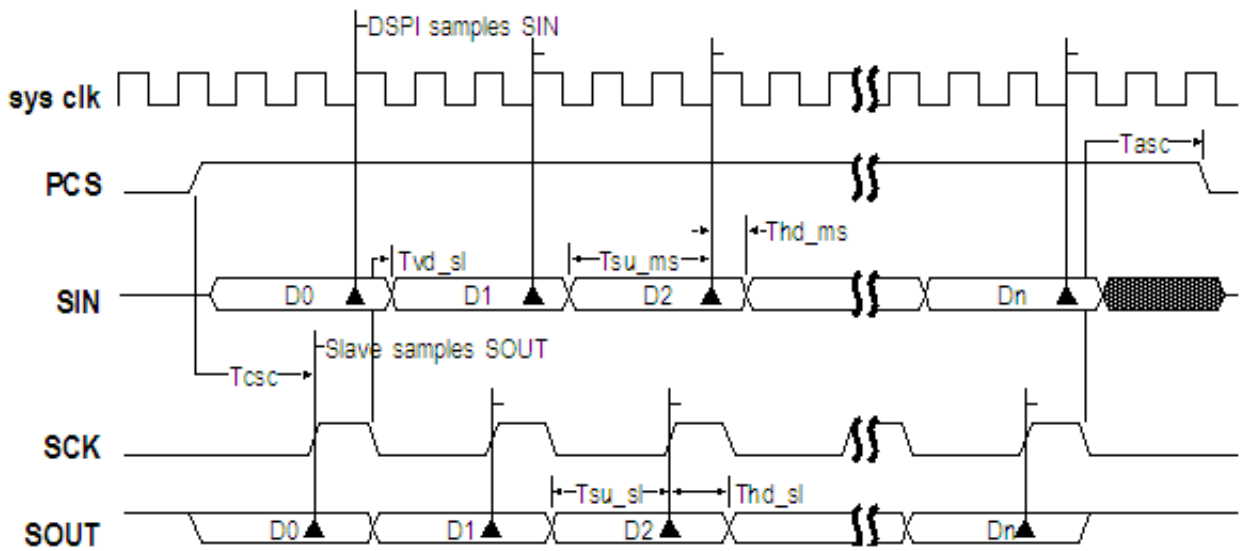


Figure 57-10. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

57.5.4.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK . The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT

signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

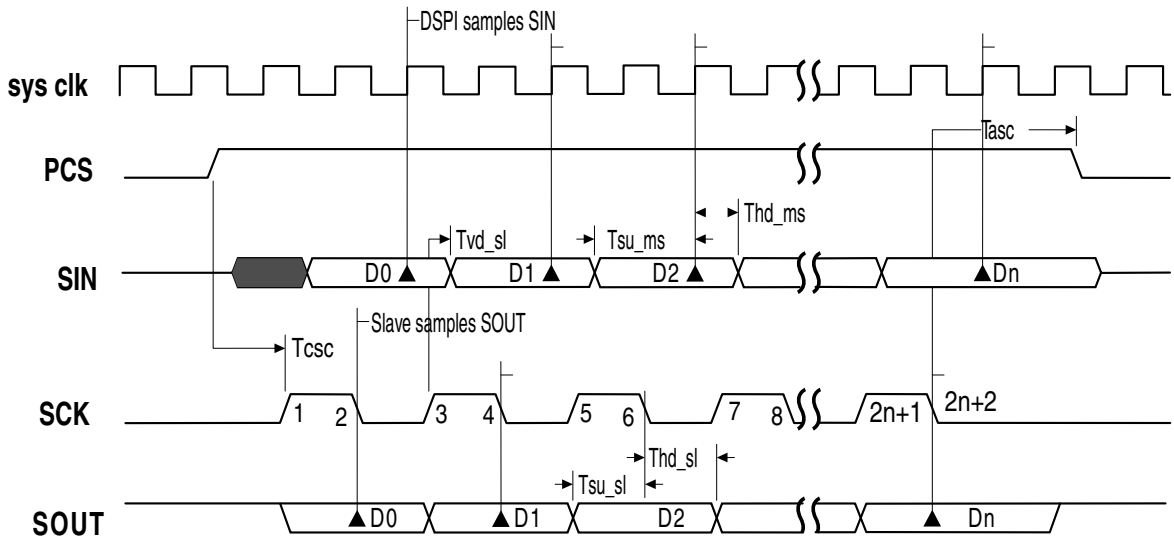


Figure 57-11. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

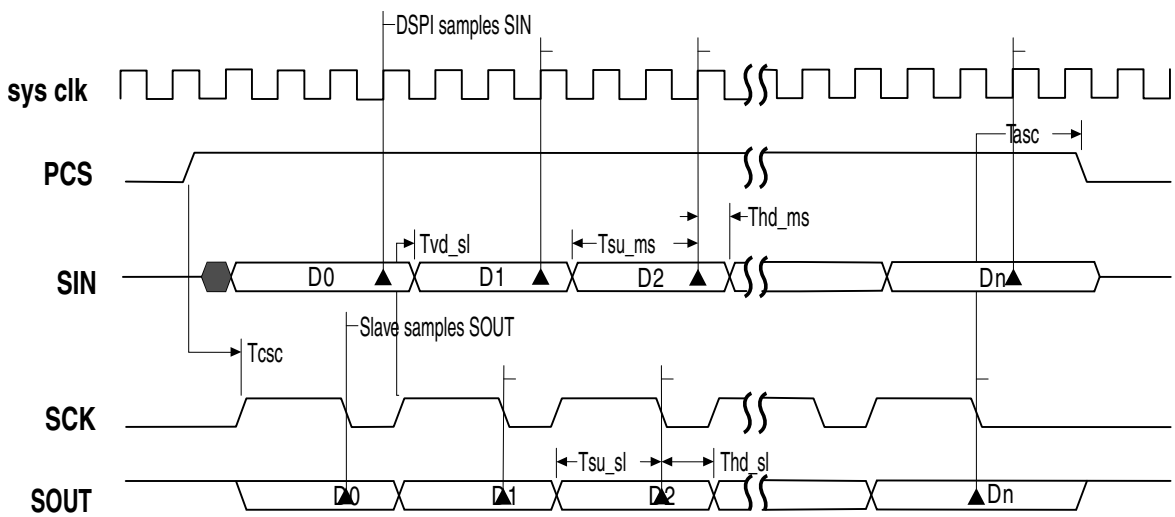


Figure 57-12. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

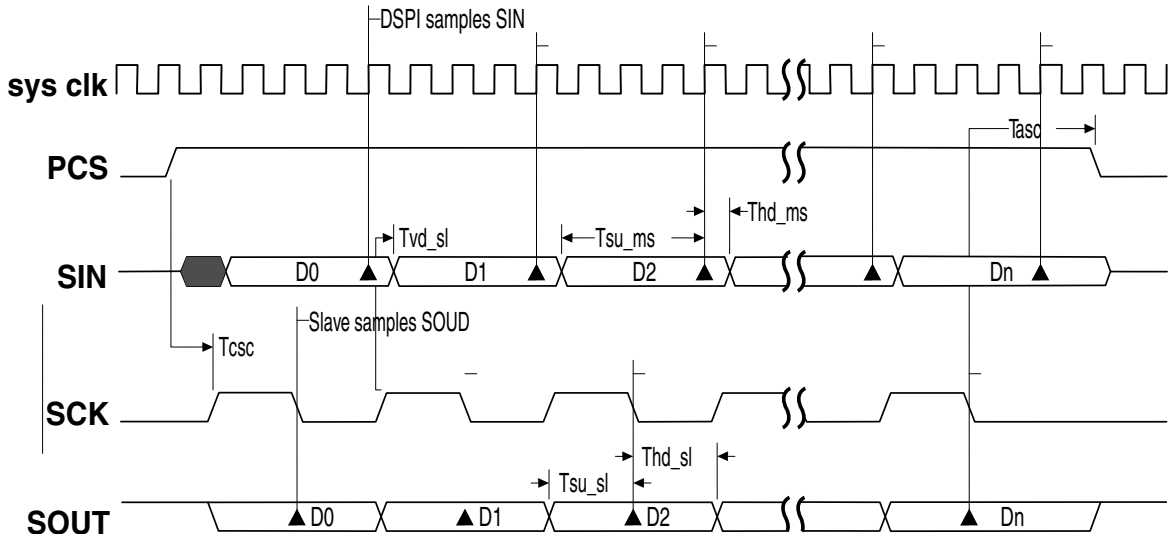


Figure 57-13. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

57.5.4.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

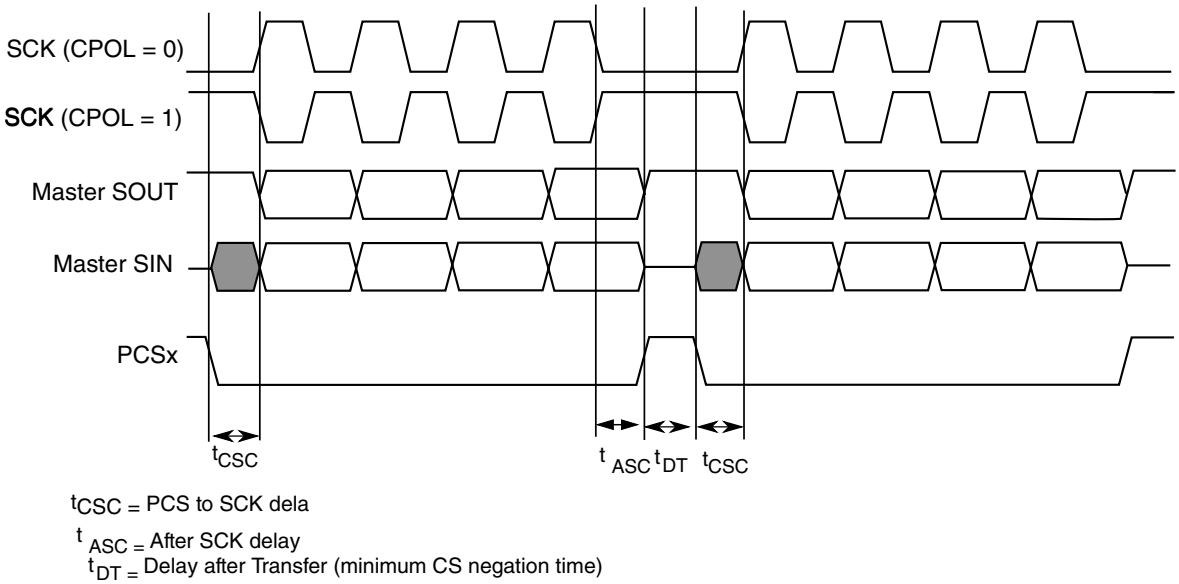


Figure 57-14. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

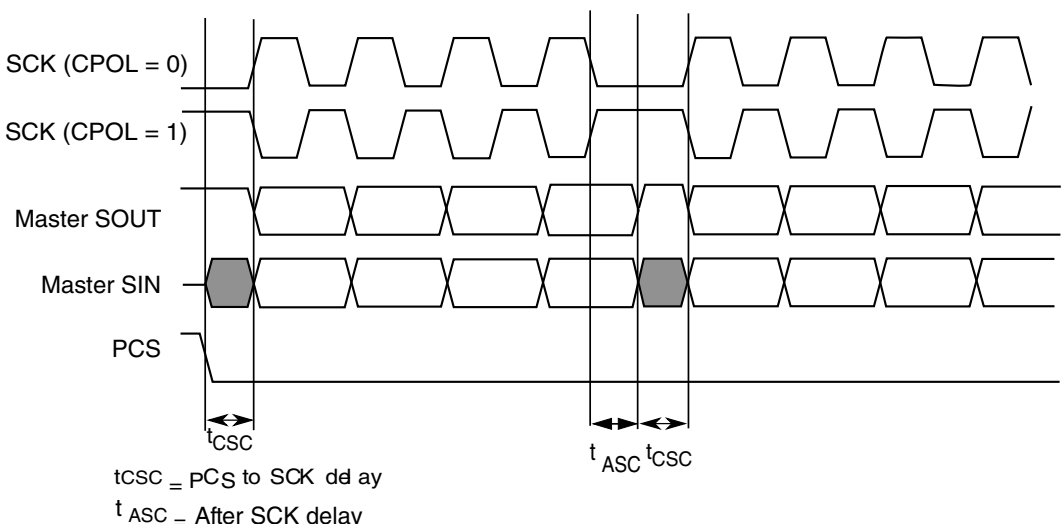


Figure 57-15. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

57.5.4.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

NOTE

The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed if the transfer is non-continuous.

The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR register. When this bit is asserted, MASC and MCSC bits of the PUSHR register perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and to speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR register.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here too.

The following figure shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

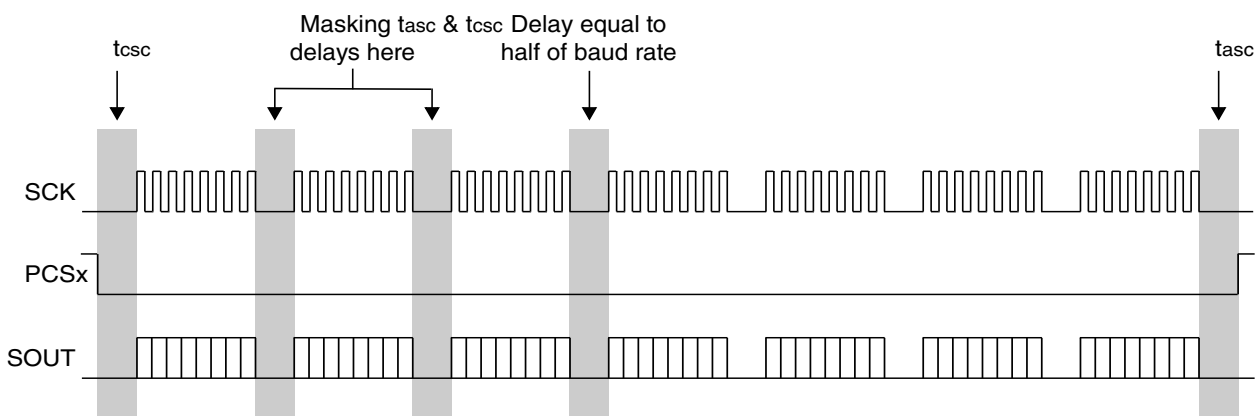


Figure 57-16. Example of Fast Continuous Selection Format

In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

57.5.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

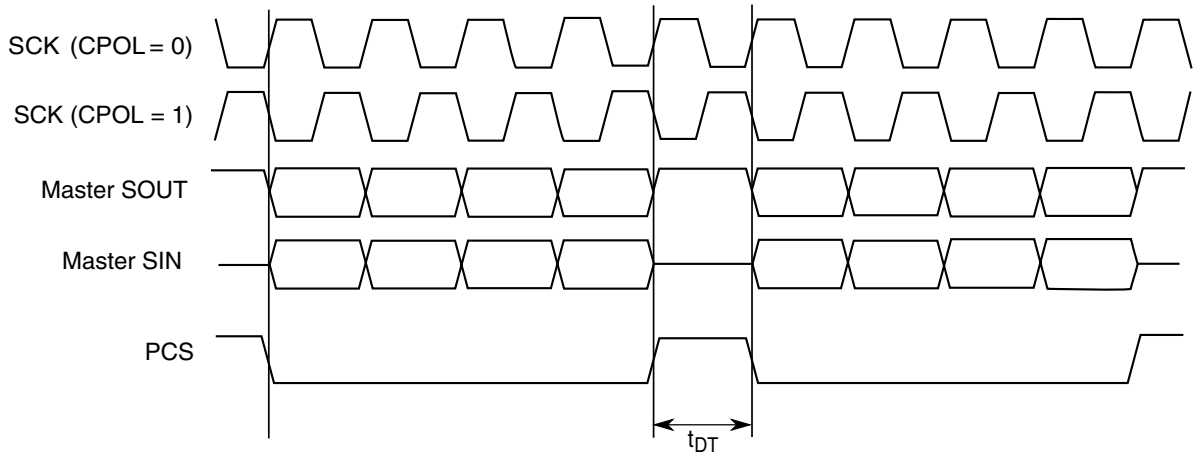


Figure 57-17. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

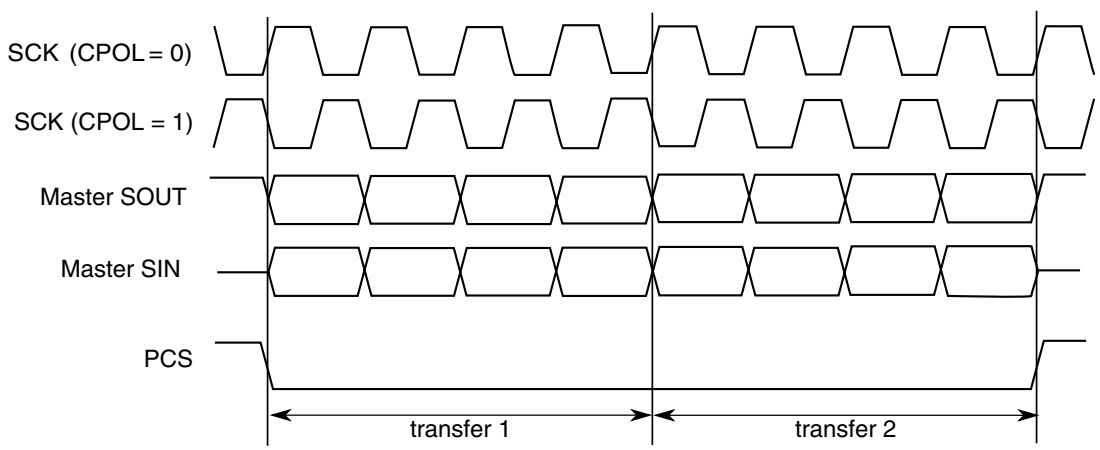


Figure 57-18. Continuous SCK timing diagram (CONT=1)

57.5.6 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

57.5.7 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/ control is done on frame basis. The registers field setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop frame transmission in case of a frame reception with parity error.

57.5.7.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

57.5.8 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 57-14. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
Transfer Complete	TCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

57.5.8.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

The module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

57.5.8.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

57.5.8.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

57.5.8.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO

of the module is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

57.5.8.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

57.5.8.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

57.5.8.7 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

57.5.9 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

57.5.9.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

57.5.9.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

57.6 Initialization/application information

This section describes how to initialize the module.

57.6.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

57.6.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].

2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

57.6.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

57.6.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

Table 57-15. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

57.6.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 57-16. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

57.6.6 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates

the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

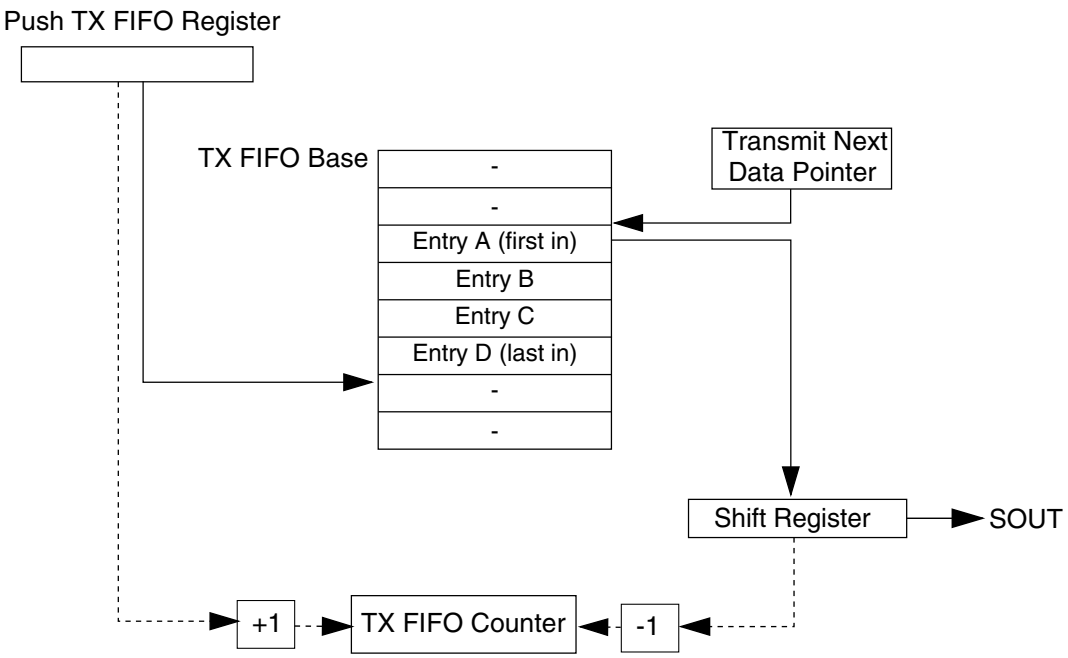


Figure 57-19. TX FIFO pointers and counter

57.6.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times \left(\text{TXCTR} + \text{TXNXTPTR} - 1\right) \bmod \left(\text{TXFIFOdepth}\right)$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

57.6.6.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO
CMDCTR - CMD FIFO Counter
CMDNXTPTR - Command Next Pointer
CMD FIFO Depth - Command FIFO depth, implementation specific

57.6.6.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 58
LINFlexD

58.1 Chip-specific LINFlexD information

58.1.1 LinFlexD Configuration

The following table shows the LinFlexD configuration.

Description	LINFlexD_1
Number of implemented filters	16
RX_CH_NUM	Number of DMA Rx channel - 1
TX_CH_NUM	Number of DMA Tx channel - 1
LIN operation mode	master/slave
Autosynchronization	Yes
Number of Tx DMA channels	1
Number of Rx DMA channels	1
DMA Support	Yes

58.2 Introduction

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, 2.1 , and 2.2. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/ UART operating mode.

58.2.1 Glossary and acronyms

Table 58-1. Glossary and acronyms

Term	Description
LIN	Local interconnect network
UART	Universal asynchronous transmitter receiver
ID	LIN identifier field
DMA	Direct memory access
TCD	Transfer control descriptor
IPS	Peripheral Bus Interface
IRQ	Interrupt request
FSM	Finite state machine
WS	Wait state
SW	Software
CPU	Central processing unit
SoC	System on a chip

58.2.2 References

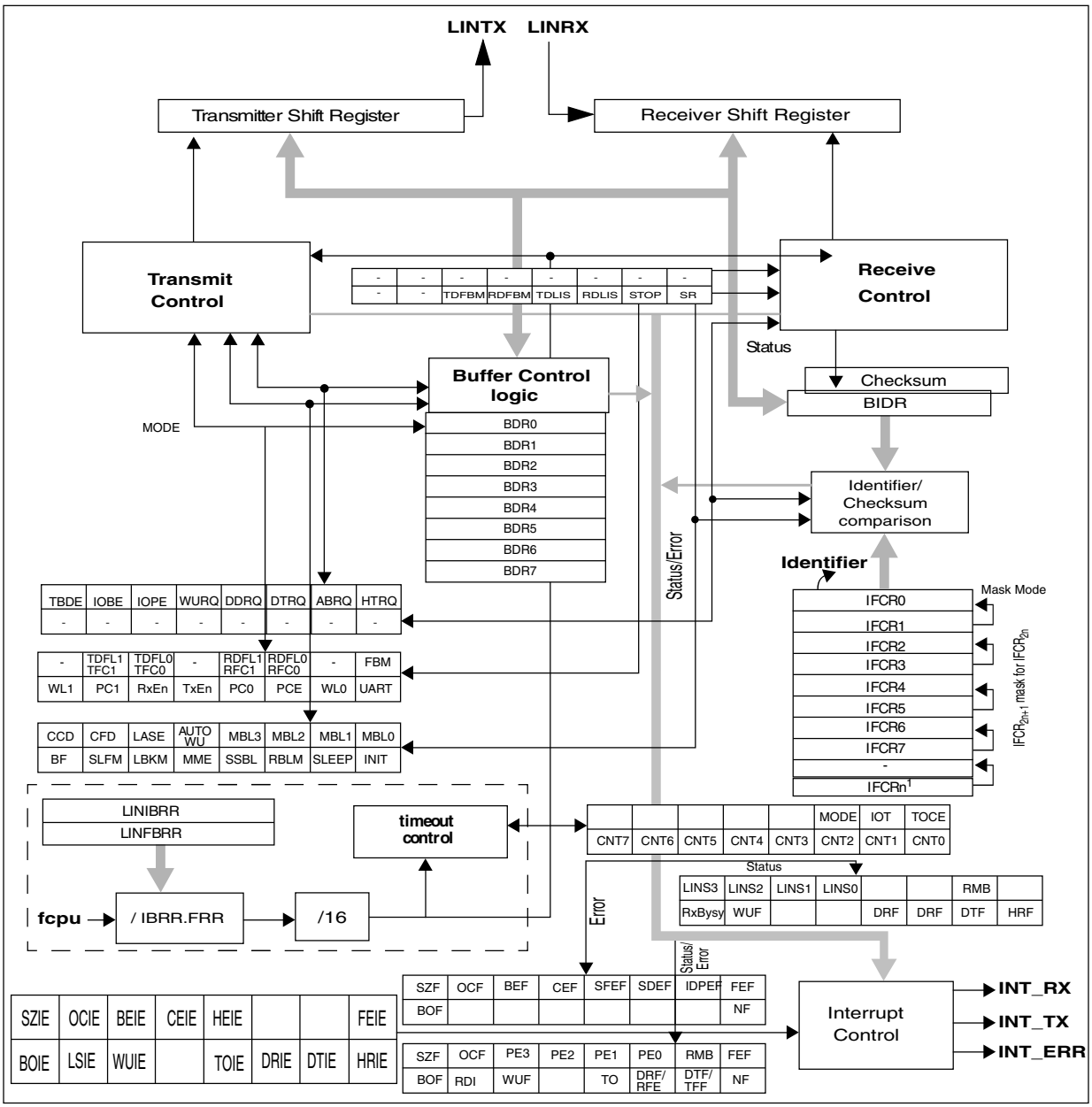


Figure 58-1. Block diagram

Notes:

1: The value of n depends on the no_of_filters. Refer to the chip configuration details to see the number of filters used in this device.

2: For LIN mode N = 16 and N is programmable by user in UART mode.

58.3 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features that are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test Mode:Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

58.3.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, 2.1, and 2.2
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Wakeup event on dominant bit detection
- True LIN field state machine

- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.

58.3.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers - see Baud rate generation
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
  - Sixteen times oversampling
  - 3:1 majority voting
- For 6.25 Mbit/s < bit rate ≤ 12.5 Mbit/s

- Reduced over sampling programmable by the user
- 3:1 majority voting for reduced over sampling of 8

- For 12.5 Mbit/s < bit rate ≤ 25 Mbit/s
  - Reduced over sampling programmable by the user
  - 1:1 voting for all reduced over sampling of 4, 5 and 6

58.4 Functional description

58.4.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

58.4.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

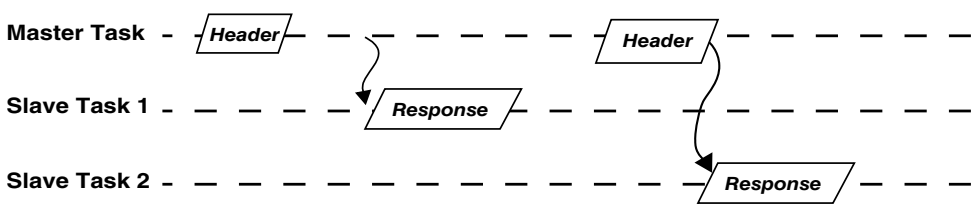


Figure 58-2. Frames

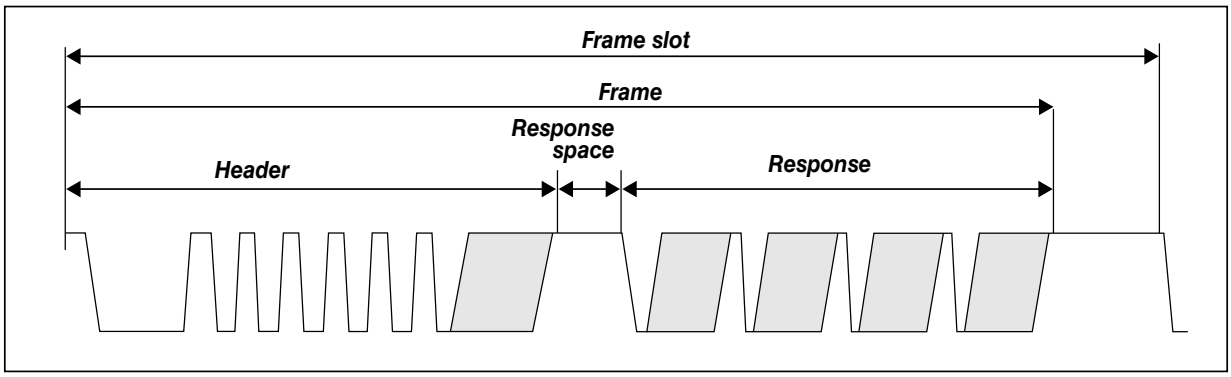


Figure 58-3. Structure of LIN frame

58.4.1.2 Data field

Each byte is transmitted as shown in Figure 58-4. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

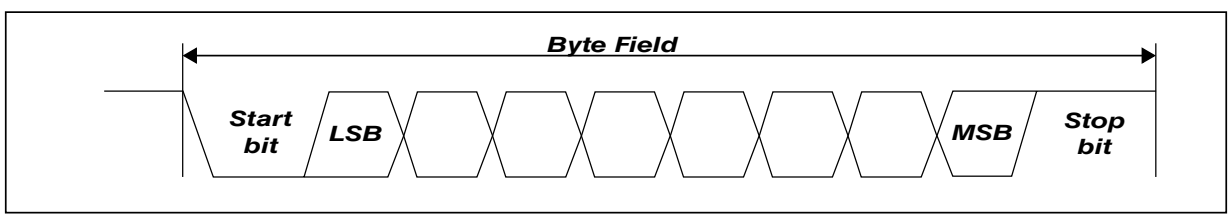


Figure 58-4. Structure of byte field

58.4.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in Figure 58-5. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

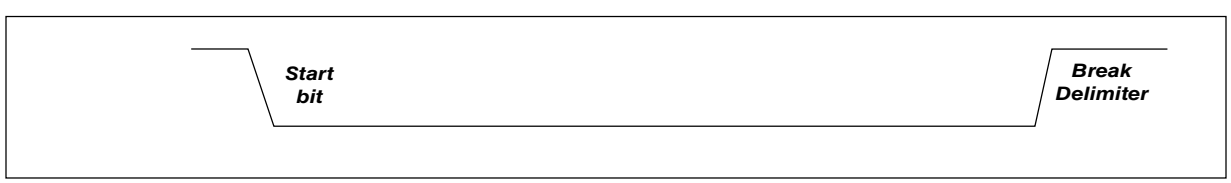


Figure 58-5. Break field

58.4.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

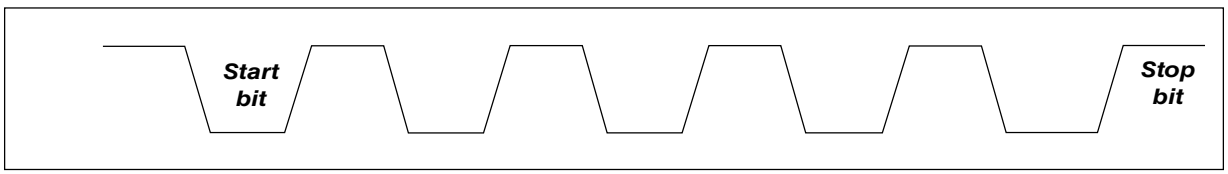


Figure 58-6. Sync byte field

58.4.1.5 Identifier

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

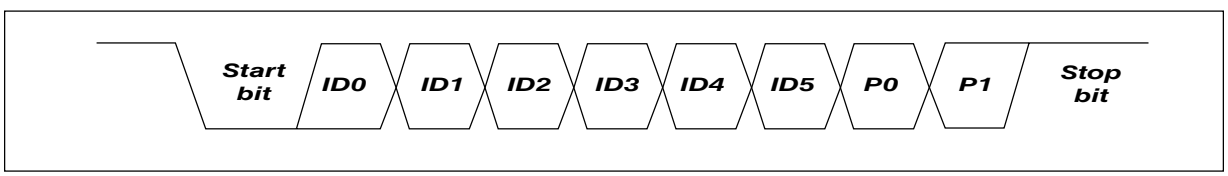


Figure 58-7. Identifier field

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

58.4.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and is used for communication with LIN 2.0 slaves.

58.4.2 LINFlexD features

58.4.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller software must:

1. Set up the baud rate registers
2. Reset UART bit
3. Select the mode (master or slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

Normal mode (NM)

Once software has completed initialization of the LINFlexD controller, it can enter the Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be awakened from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to Wakeup management.

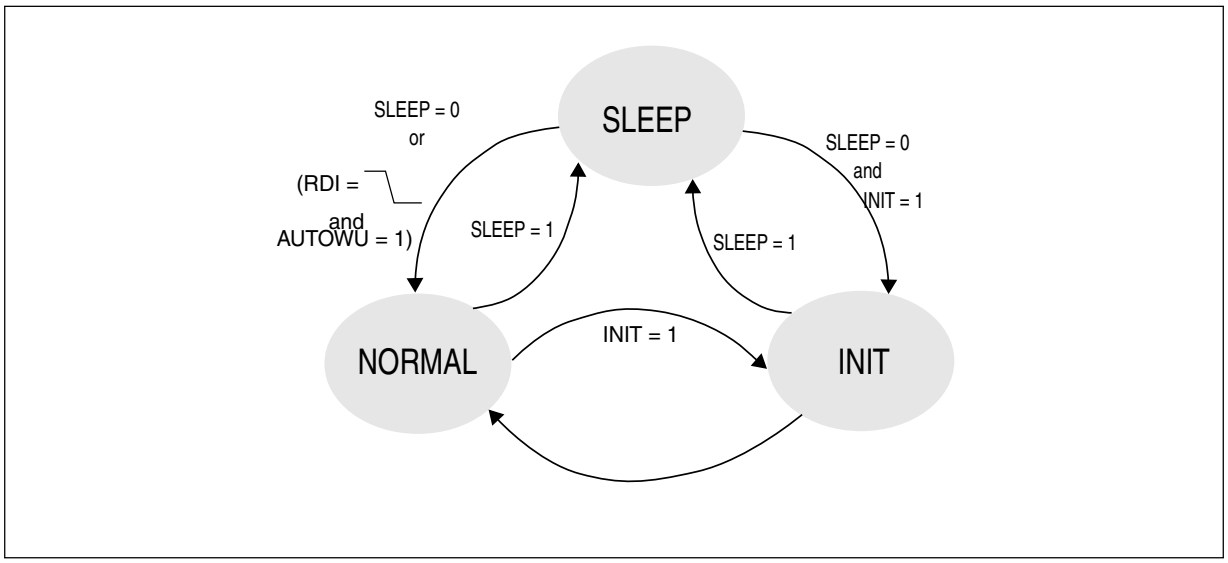


Figure 58-8. Operating modes

58.4.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is an internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

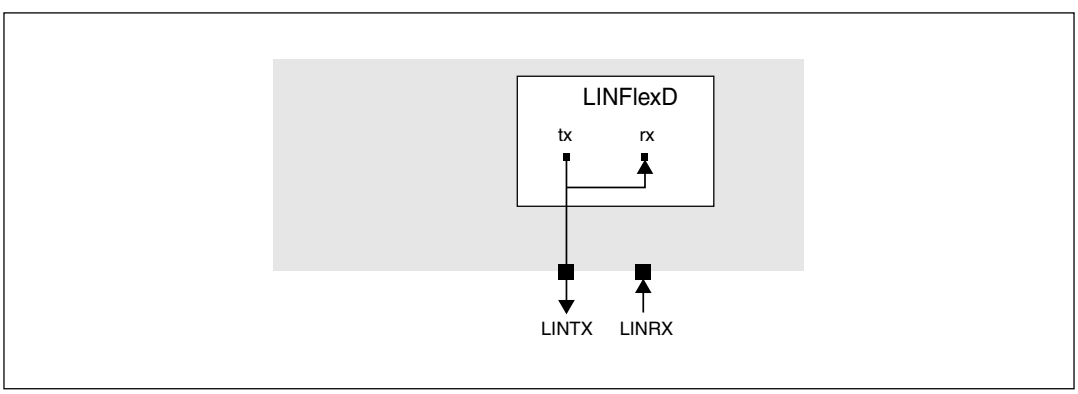


Figure 58-9. LINFlexD in Loop Back Mode

58.4.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDRL and BDRM. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling).

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Error handling).

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

58.4.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit, and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7].

The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX

interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software then you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

58.4.2.5 Errors

58.4.2.5.1 Header error

A header error is an error during the header reception. The error types are:

1. Sync Del error (SDEF)
2. Sync field error (SFEF)
3. Identifier Parity error (IDPEF)

Sync Del error (SDEF)

The delimiter should be 1 for at least one bit time; otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if *HEIE* bit of LINEIER is set.

Sync field error (SFEF)

The SFEF error condition is monitored differently depending on whether autosynchronization (LASE) is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the SFEF flag indicates:

- The deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators or

- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.
- Doesn't indicate that an inconsistent Sync Field other than 0x55 has been received by the slave.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator).

This check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If SFEF field is asserted it means that:

D > 14.0625%

If there is no error, it means that:

D < 14.84375%

If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

Overflow during Sync Field Measurement

This check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = '0')

In this case the Sync character is received as a normal character. If the received character is 0x55 then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

Identifier Parity error(IDPEF)

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

58.4.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CPU cycles. Bit error is not checked during break field transmission.

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) - (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note

If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

58.4.2.5.3 Framing error

This error is flagged when a dominant state is sampled on stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF is never set in this case.

58.4.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

58.4.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of slave, if buffer is not released (RMB is not reset) before reception of next Identifier and if RBLM is set, then the ID along with the data is discarded.

58.4.2.5.6 Timeout error

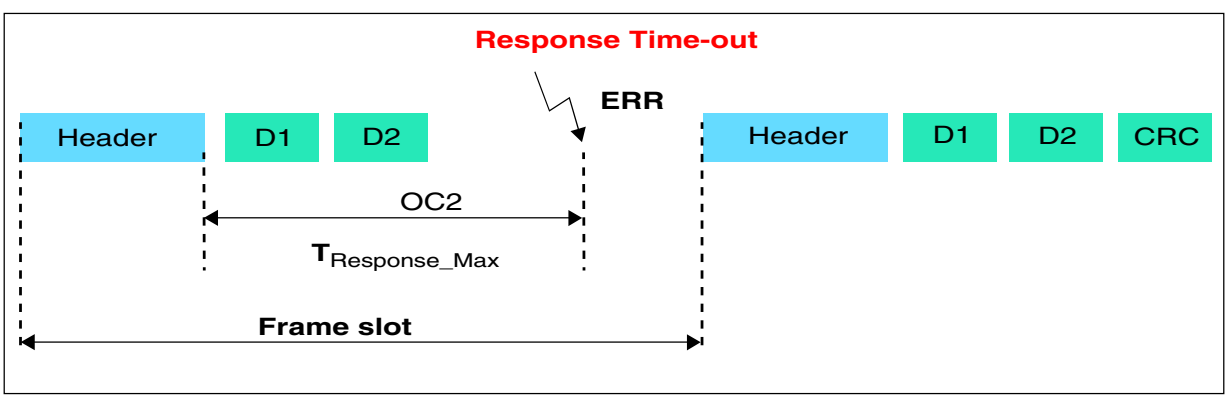


Figure 58-10. Incomplete response (for example, missing checksum)

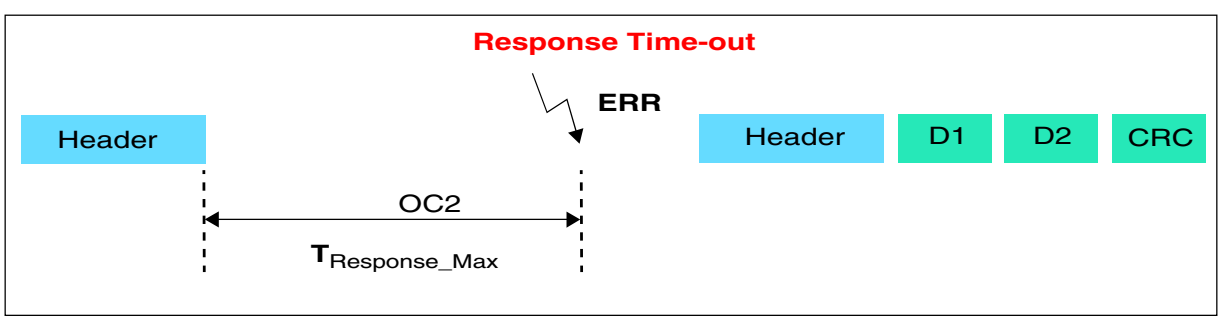


Figure 58-11. No response

Response timeout mechanism

- Master mode
  - OC2 is loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time).

    This loading takes place at the end of id field (and not at the beginning of data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further OC2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 value (no change compared to earlier implementation).
- Slave mode
  - **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, OC2 can be loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time))
  - **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the implementation is the following:
    - At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT].
    - At the end of the first_data_byte it is reloaded again according to DFL
    - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

NOTE

The value of nominal_time_out (1.4 × ((DFL +2) × 10 bit time)) shown is as per the reset value of LINTOCR[RTO] and LINCR1[CFD]. It changes with change in value of LINTOCR[RTO] and LINCR1[CFD]. Here, 1.4 corresponds to (LINTOCR[RTO])/10 and 2 corresponds to (2- LINCR1[CFD])

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length),
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit
  - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit
  - If the counter starts after 11 Tbit (break duration), the HTO value is 56 –11 = 45 Tbit.

  The reset value of HTO is 45, and this register can only be programmed in slave mode. Counter restarts after break duration and OC1 gets loaded with the value of HTO.

  As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

  The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be reset and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

58.4.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8^{th}, 9^{th} and 10^{th} samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see LIN Error Status Register (LINFlexD_LINESR).

58.4.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2n and 2n+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/ reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2n or IFCR2n+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2n is the filter and 2n+1 acts as a mask for it. The FACT bit for filter 2n+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

If there is a match of identifier with the mth filter in any mode, then m+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the

HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note

If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In mask mode if two filters match with the identifier then the filter having the lower number prevails.

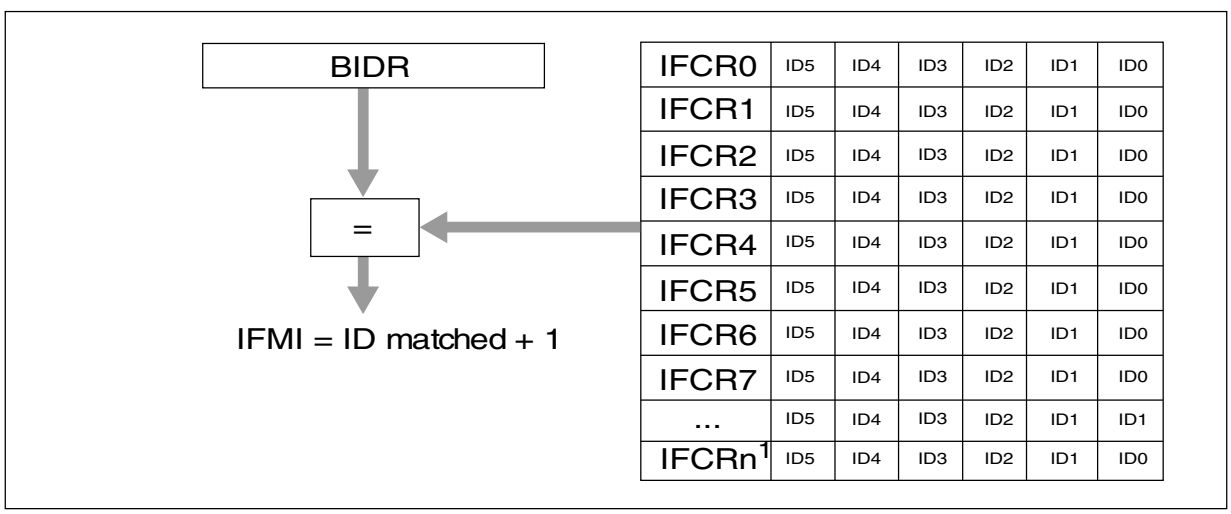


Figure 58-12. Identifier List mode

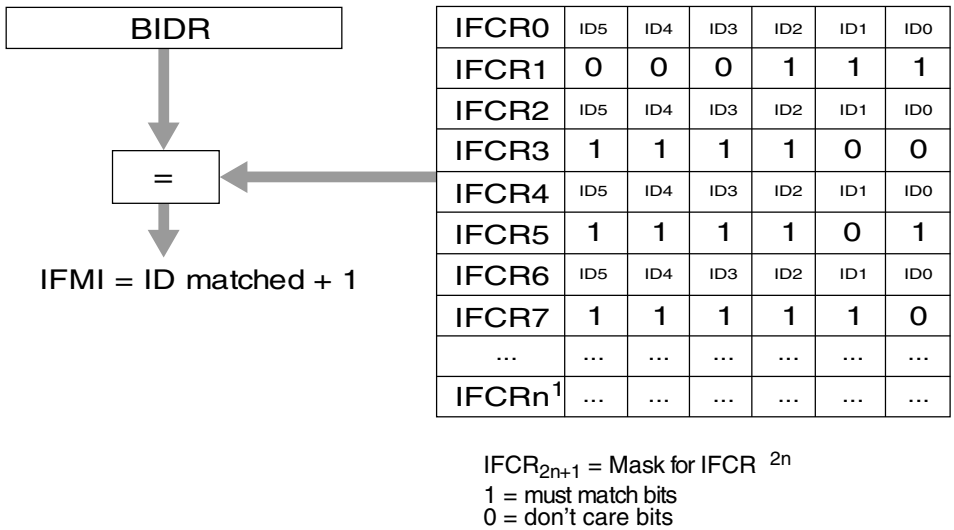


Figure 58-13. Identifier Mask mode

58.4.2.7 Start detection and break delimiter detection in receiver

There is a 10-bit-shift register sample register in the receiver that shifts signal data to the next least significant bit on each incoming sample.

Table 58-2. Start Detection and Delimiter Detection in receiver

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13

Table continues on the next page...

Table 58-2. Start Detection and Delimiter Detection in receiver (continued)

9	8	7	6	5	4	3	2	1	0	Sample_reg
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x	0	x	0	x	0	start detect
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	delimiter detect
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

58.4.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to Table 58-2 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘0’, then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.

6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

58.4.2.7.2 Break delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to Table 58-2 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘1’, then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

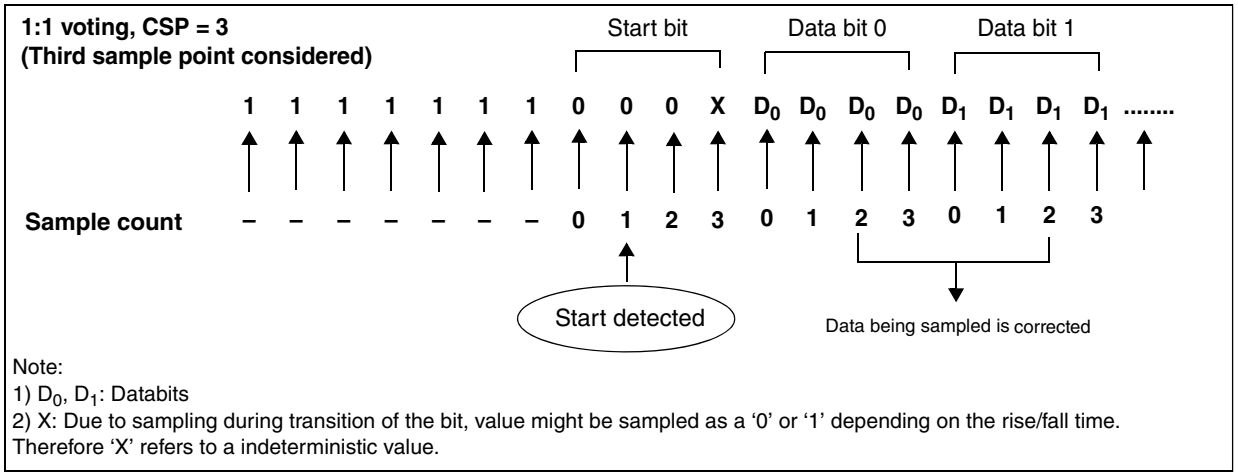


Figure 58-14. Start detection and sampling for over sampling rate = 4 (Case 1)

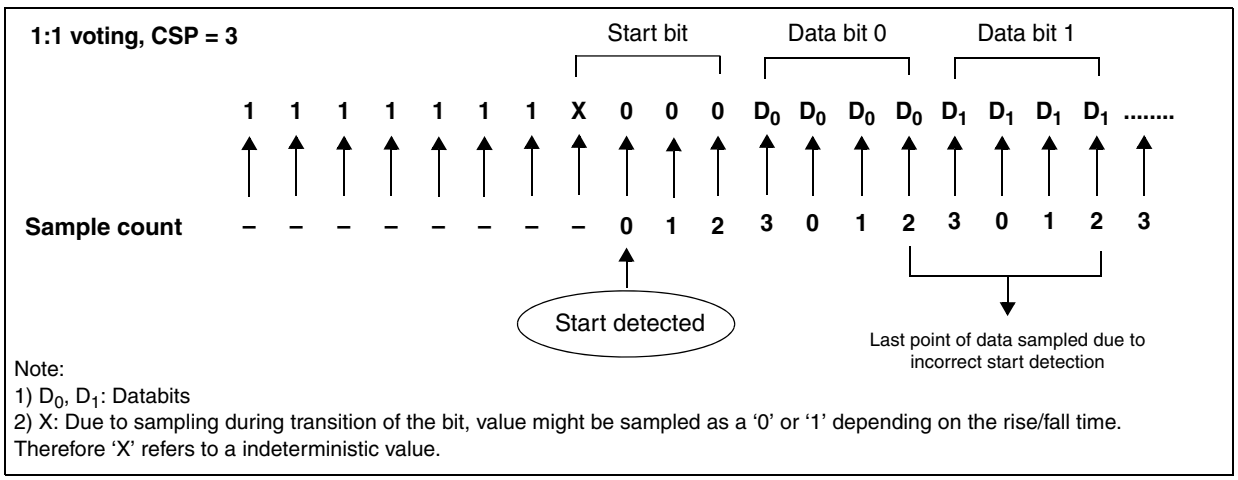


Figure 58-15. Start detection and sampling for over sampling rate = 4 (Case 2a)

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

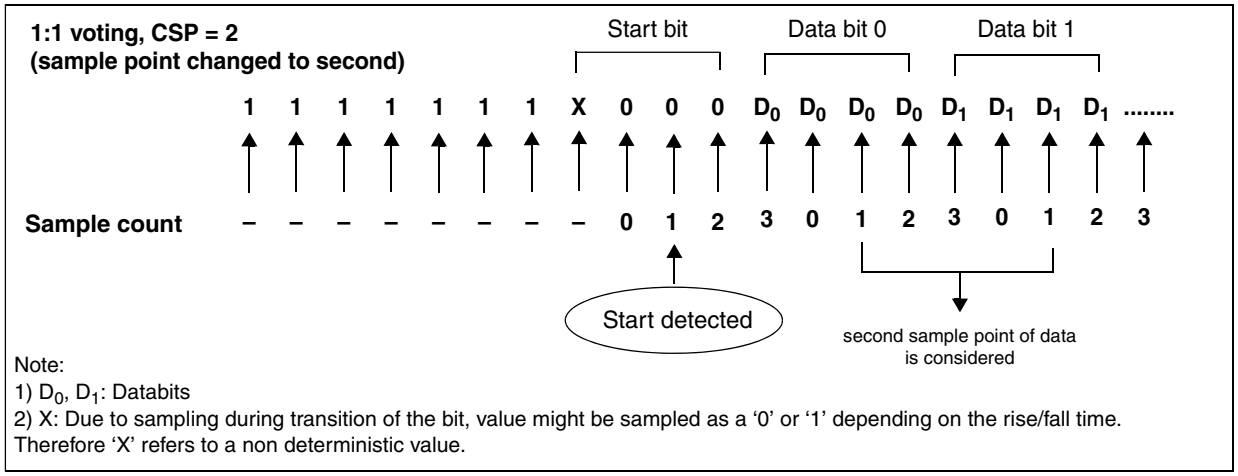


Figure 58-16. Start detection and sampling for over sampling rate = 4 (Case 2b)

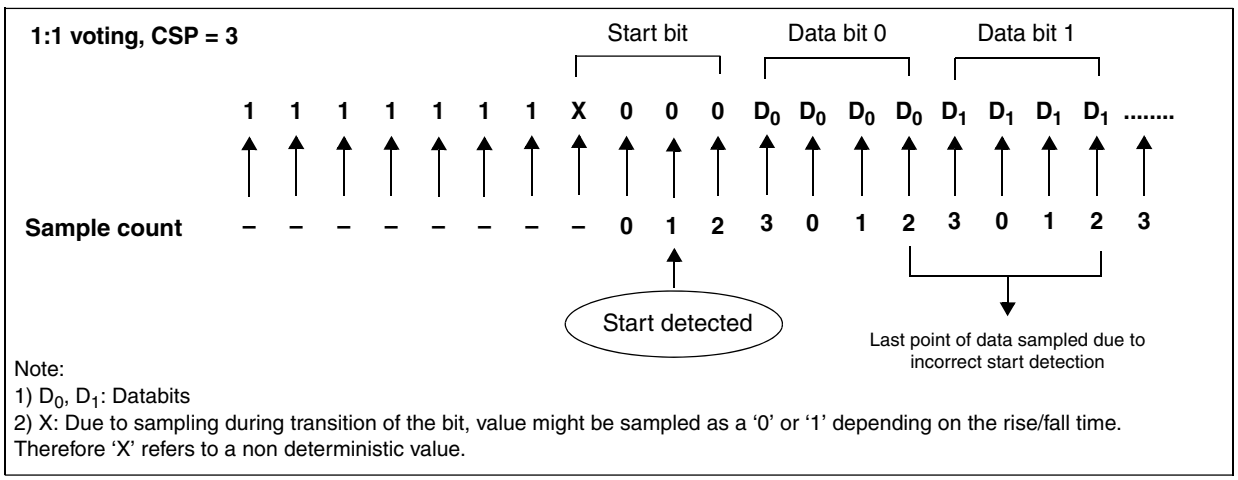


Figure 58-17. Start detection and sampling for over sampling rate = 4 (Case 3)

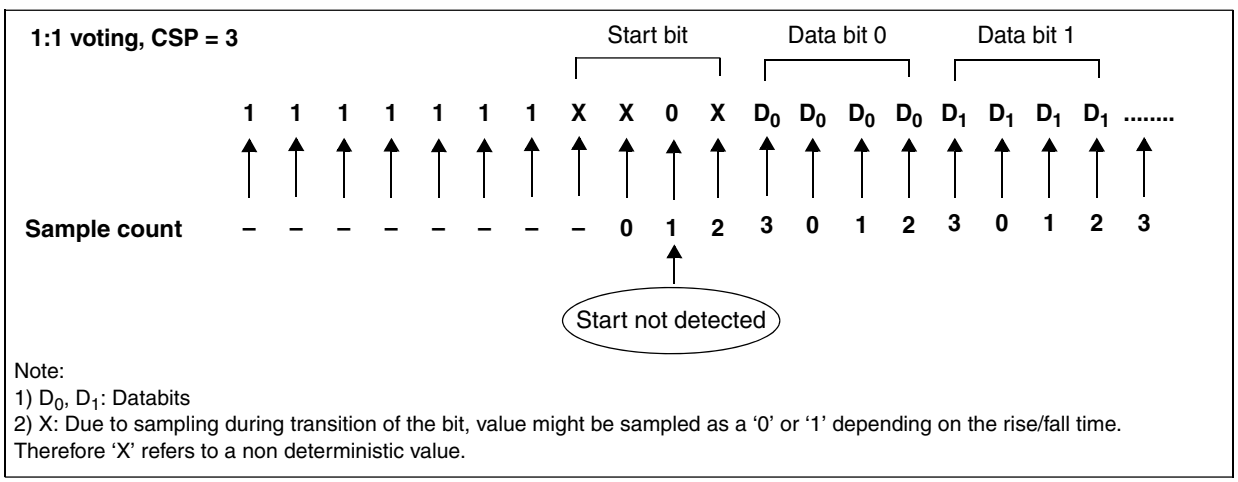


Figure 58-18. Start detection and sampling for over sampling rate = 4 (Case 4)

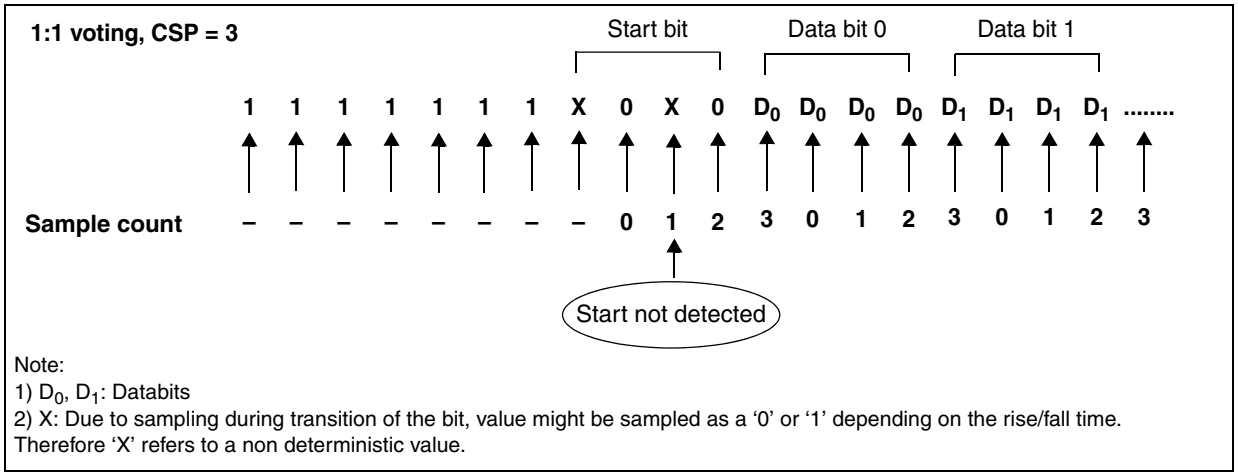


Figure 58-19. Start detection and sampling for over sampling rate = 4 (Case 5)

58.4.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate Register and the LIN Fraction baud rate register. The Baud Rate Registers can be programmed only during Initialization mode.

Baud rate is calculated with the following formula for both receiver and transmitter.

When ROSE = 1, Tx = Rx = LIN_CLK / (OSR × LDIV). When ROSE = 0, Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock.

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled, LINFBRR should not be used and programmed to zero and LDIV contains only the integer part of LINIBRR.

For example: When ROSE = 0 (For LIN and UART mode): LDIV = 468.75 d, LIN_CLK = 36 MHz, LINIBRR = 468 d, LINFBRR = 12 *Baud rate = 36 MHz/ (16 × 468.75) = 4.8 Kbit/s*

For example: When ROSE = 1 (Only for UART mode): LDIV = 10 d, LIN_CLK = 80 MHz, LINIBRR = 4 d, OSR = 4, *Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s*

58.4.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescalar value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When auto synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

58.4.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 μs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 μs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1 On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated if WUPIE is set. LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0 On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE bit is set). It is then up to the software to clear the SLEEP bit.

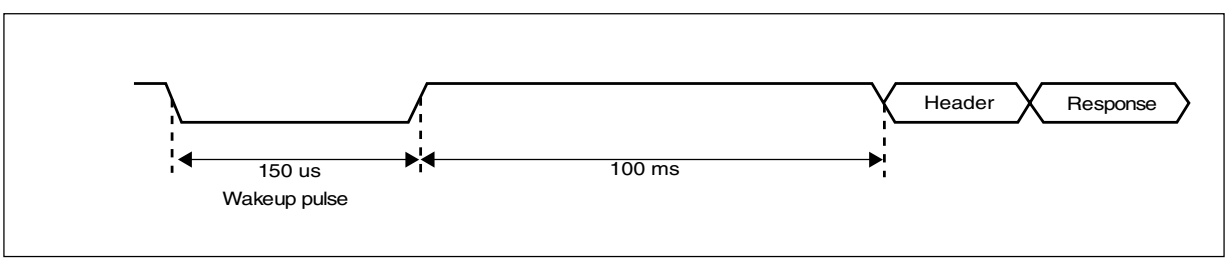


Figure 58-20. Wakeup sequence

58.4.3 Timer

There is an 8-bit counter which has different behavior in different modes as described below:

- **In Output Compare Mode (LINTCSR[MODE]= 1):**

  This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has reset the stuck FSM state.
- **In LIN Mode:**

  The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

58.4.4 UART mode

Main features in the UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 13-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

58.4.4.1 8-bit data frames

The eighth bit can be a data or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

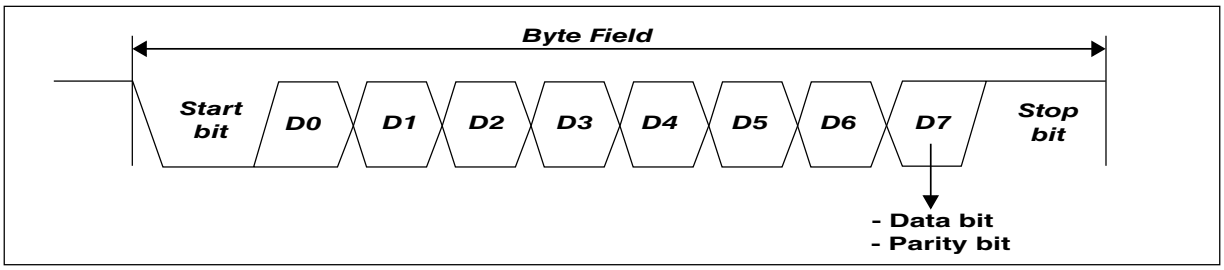


Figure 58-21. UART mode 8-bit data frame

58.4.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

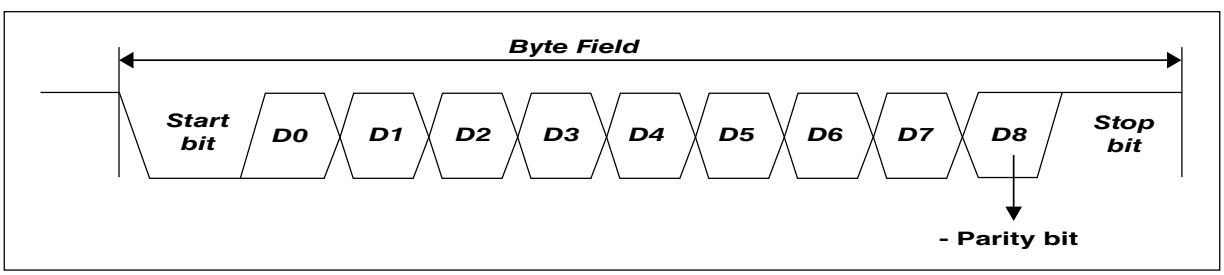


Figure 58-22. UART mode 9-bit data frame

58.4.4.3 16-bit data frames

The sixteenth bit can be a data or a parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

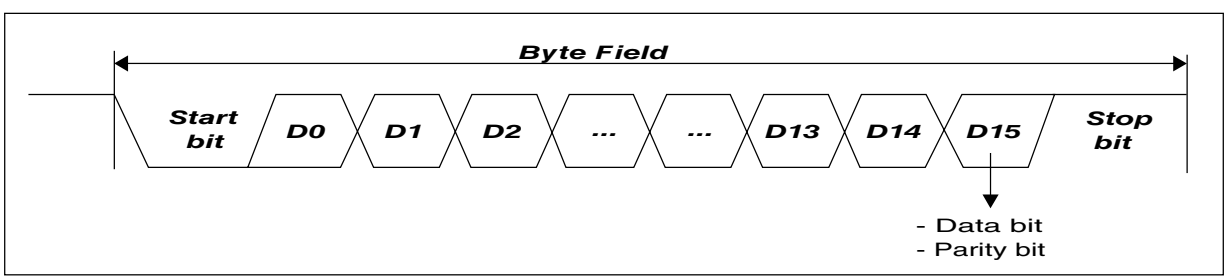


Figure 58-23. UART mode 16-bit data frame

58.4.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

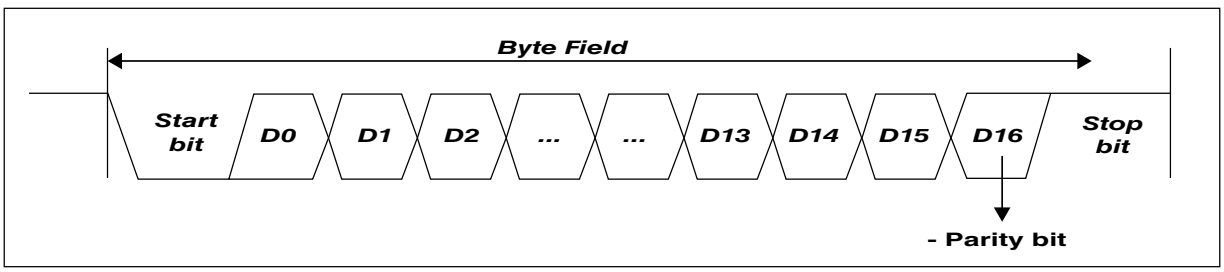


Figure 58-24. UART mode 17-bit data frame

58.4.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

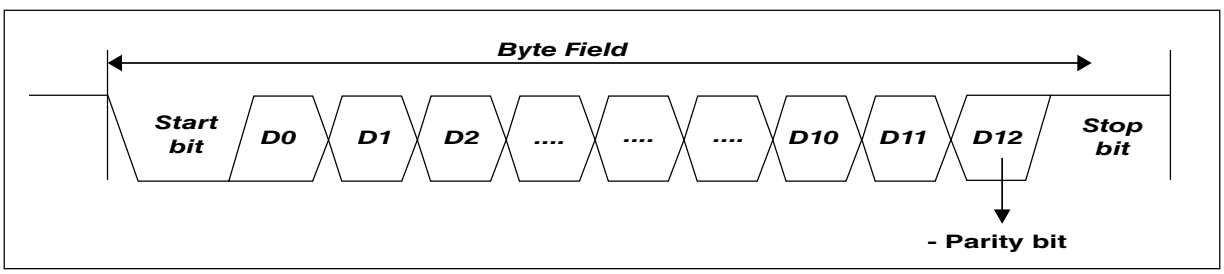


Figure 58-25. UART mode 13-bit data frame

58.4.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in the following figure:

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

Figure 58-26. Structure of 8-byte buffer

For 16-bit frames, the lower eight bits are written in BDR0 and upper eight bits in BDR1.

The same applies for reception.

58.4.4.7 UART transmitter

UART transmitter

In order to start transmission in the UART mode, UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the TxEn bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

NOTE

If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 58-3. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Byte0	BDRL	FIFO	Byte	OK
Write Byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write Half-word0-1	BDRL	FIFO	Byte	IPS transfer error
Write Word	BDRL	FIFO	Byte	IPS transfer error
Write Byte0-1-2-3	BDRL	FIFO	Half-word	IPS transfer error
Write Half-word0	BDRL	FIFO	Half-word	OK
Write Half-word1	BDRL	FIFO	Half-word	IPS transfer error
Write Word	BDRL	FIFO	Half-word	IPS transfer error
Read Byte0-1-2-3	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Half-word0-1	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Word	BDRL	FIFO	Byte/Half-word	IPS transfer error
Write Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Write Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 58-3. BDRL access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Word	BDRL	BUFFER	Byte/Half-word	OK
Read Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Read Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Read Word	BDRL	BUFFER	Byte/Half-word	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

58.4.4.8 UART receiver

UART receiver

The reception of a data byte is started as soon as the user exits initialization mode, sets the RxEn bit, and detects start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-half-words (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. RxEn bit needs to be set only to start the reception. The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 58-4. BDRM access in UART mode

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Byte4	BDRM	FIFO	Byte	OK
Read Byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read Half-word2-3	BDRM	FIFO	Byte	IPS transfer error
Read Word	BDRM	FIFO	Byte	IPS transfer error
Read Byte4-5-6-7	BDRM	FIFO	Half-word	IPS transfer error
Read Half-word2	BDRM	FIFO	Half-word	OK
Read Half-word3	BDRM	FIFO	Half-word	IPS transfer error
Read Word	BDRM	FIFO	Half-word	IPS transfer error
Write Byte4-5-6-7	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Word	BDRM	FIFO	Byte/Half-word	IPS transfer error
Read Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 58-4. BDRM access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Half-word2-3	BDRM	BUFFER	Byte/Half-word	OK
Read Word	BDRM	BUFFER	Byte/Half-word	OK
Write Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Word	BDRM	BUFFER	Byte/Half-word	IPS transfer error

Note

Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and data bits of the registers BDRL/BDRM.

Note

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number of bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to Idle state.
- If a STOP request arrives in the middle of one reception, it is only acknowledged after all the programmed number of data bytes have received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout. When the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if FEIE bit in LINIER is set. As there is only one register bit for framing error, this interrupt will be helpful in identifying which byte has framing error.

- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is depending on RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.
- When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

58.4.5 DMA interface

58.4.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channels where N = 2^TX_CH_NUM. See chip specific information for the value of TX_CH_NUM used in this device.
- LIN Slave node, RX mode: 1 to N DMA channels where N = 2^RX_CH_NUM. See chip specific information for the value of RX_CH_NUM used in this device.
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

58.4.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in Table 58-5 and Table 58-6.

Table 58-5. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL	—	Data field length: • 1 to 8 bytes: normal frames
	DIR	—	Direction of the data field: • 0 => reception • 1 => transmission
	CCS	—	Checksum type: • 0 enhanced • 1 classic
	ID[5:0]	—	Identifier.
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA. • 0 buffer data is free • 1 buffer data is ready
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received. Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode).
UARTCR	FBM	—	FIFO/Buffer mode • 0 Buffer mode • 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty • 0 Rx FIFO not empty • 1 Rx FIFO empty
	TFF	—	Tx FIFO full • 0 Tx FIFO not full • 1 Tx FIFO full
DMATXE[2**TX_CH_NUM][1]		High	DMA Tx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed. In LIN slave mode: # DMA TX channel = IFMI[3:0] -1.

Table continues on the next page...

Table 58-5. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
			DMATXE[x] = 0b => TX channel x disabled DMATXE[x] = 1b => TX channel x enabled

DMARXE[2**RX_CH_NUM][1]		High	DMA Rx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA RX channel = IFMI[3:0] -1.
			DMARXE[x] = 0b => RX channel x disabled
			DMARXE[x] = 1b => RX channel x enabled

1. ** stands for exponentiation.
1. ** stands for exponentiation

Table 58-6. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0]	High	Logical AND between the 2 DMA enable bits (LINFlexD and eDMA).
DMA_TEN	LIN slave Tx	DMATXE[x] x = IFMI — 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] x = IFMI — 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in Table 58-7.

Table 58-7. TCD control fields description

TCD Field	Level	Description
CITER[14:0]	—	Current "major" iteration count
BITER[14:0]	—	Beginning "major" iteration count
NBYTES[31:0]	—	Inner "minor" byte transfer count. Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 => 8-bit 001 => 16-bit

Table continues on the next page...

Table 58-7. TCD control fields description (continued)

TCD Field	Level	Description
		010 => 32-bit 011 => 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request.
DONE	High	Channel done (the DMA has completed the outer major loop).
D_REQ	High	Disable request. If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

58.4.5.3 Master node –TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

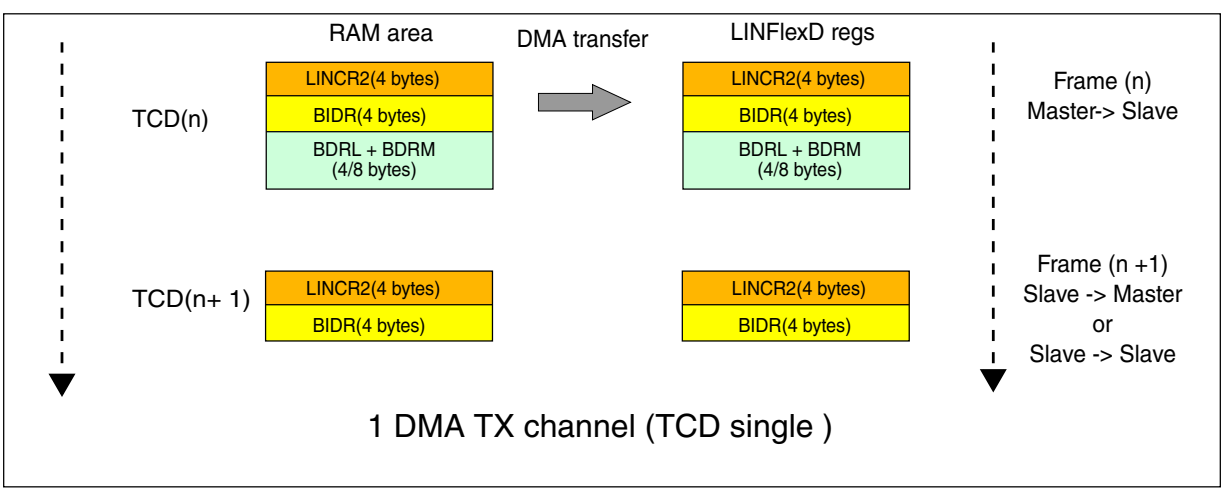


Figure 58-27. Master node –TX memory map

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

The following table provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 58-8. Master node – Tx mode – Register setting

LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in Table 58-9. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 58-9. TCD setting – Master node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	-	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer

Table continues on the next page...

Table 58-9. TCD setting – Master node – Tx mode (continued)

TCD Field	Value	Description
SLAST[31:0]	–N	
DADDR[31:0]	-	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

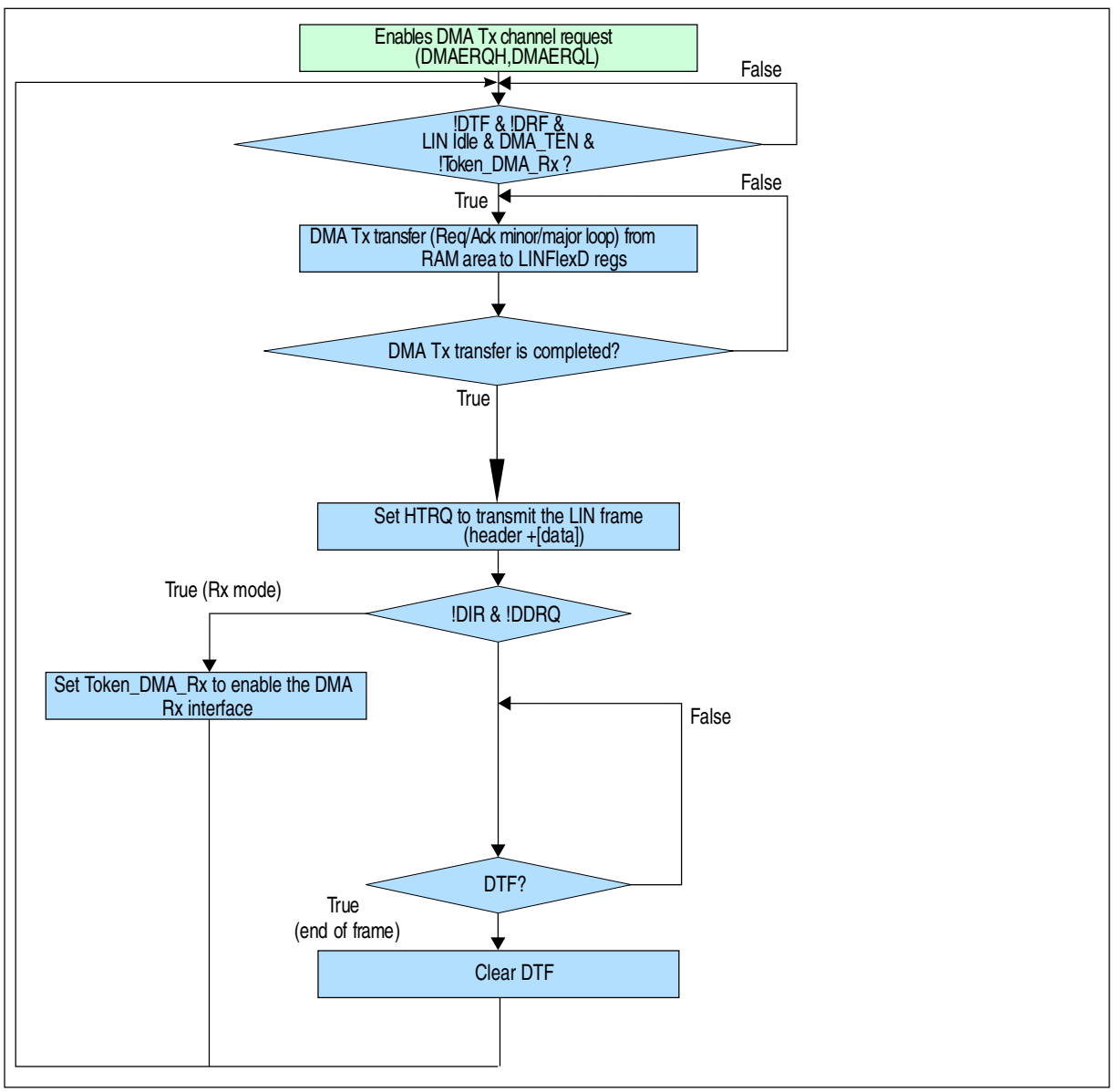


Figure 58-28. Master node – DMA Tx FSM (concept scheme)

58.4.5.4 Master node - RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame . The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

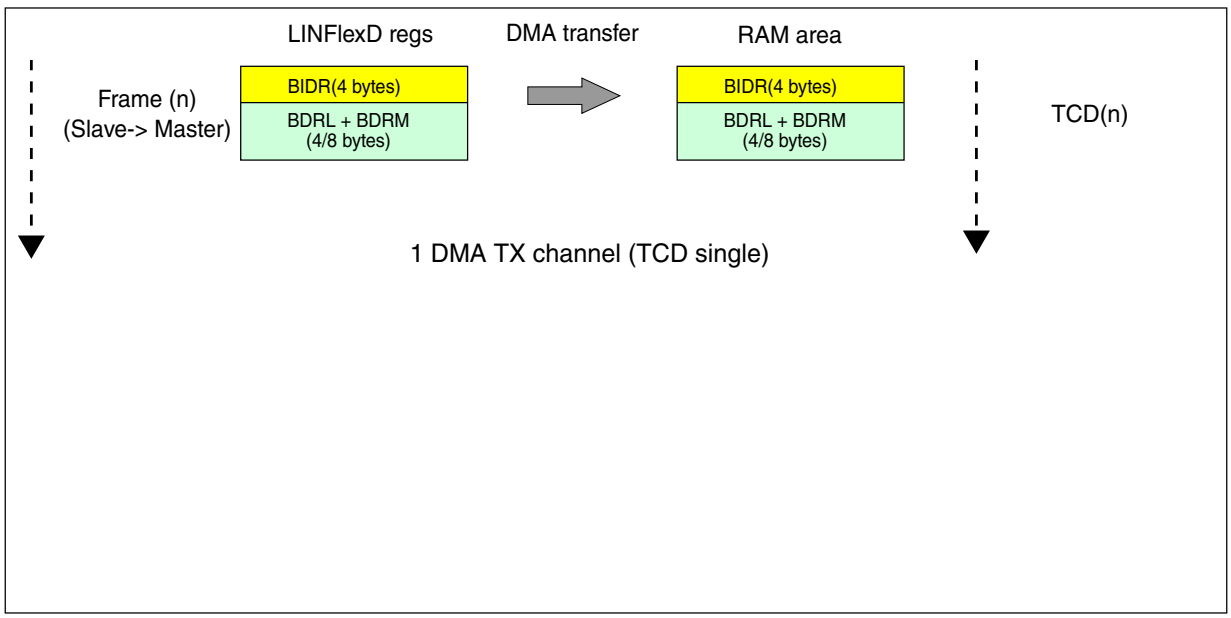


Figure 58-29. Master node – RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in the following figure. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in the next table. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 58-10. TCD setting – Master node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM

Table continues on the next page...

Table 58-10. TCD setting – Master node – Rx mode (continued)

TCD Field	Value	Description
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

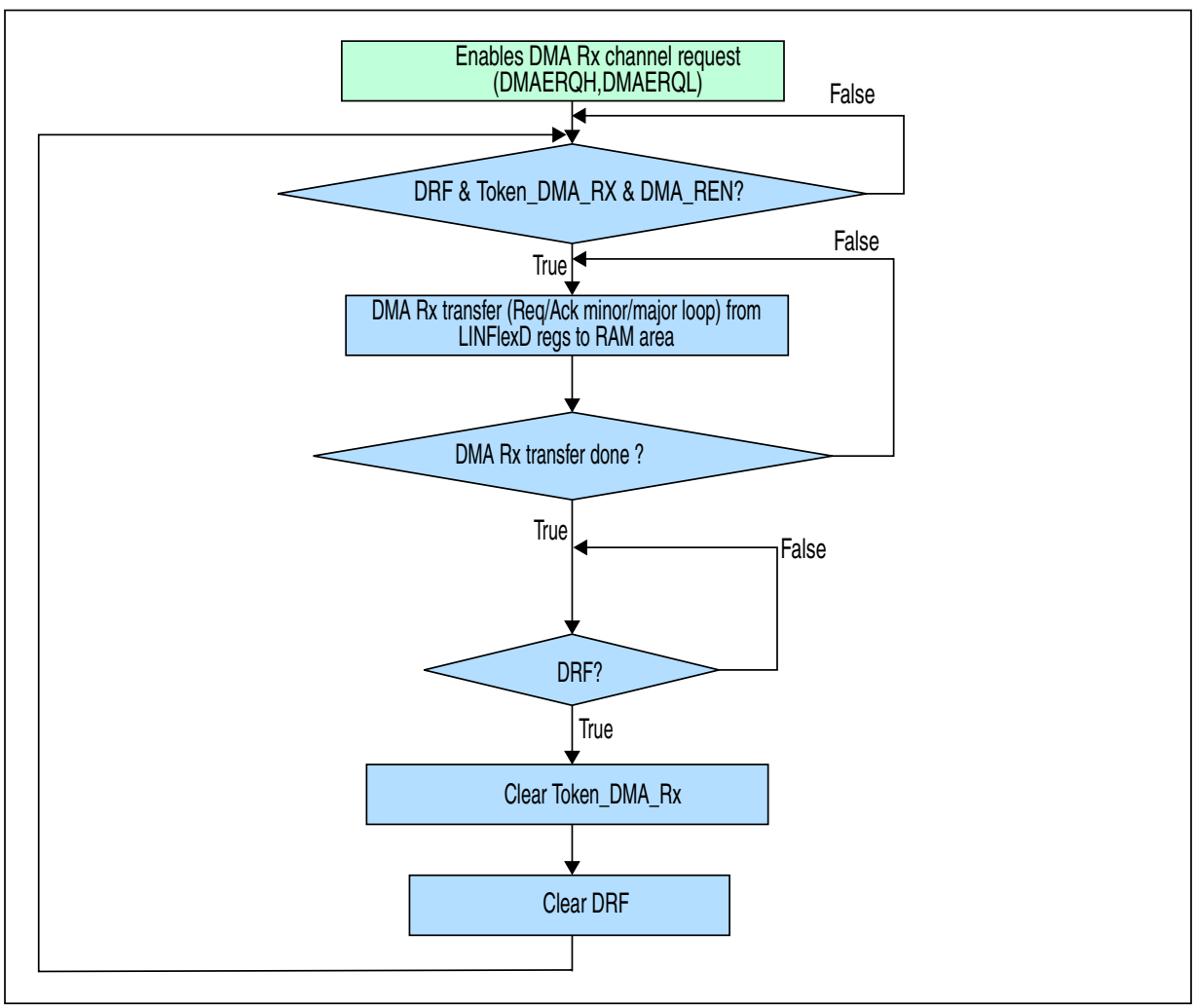


Figure 58-30. Master node – DMA Rx FSM (concept scheme)

58.4.5.5 Slave node – TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

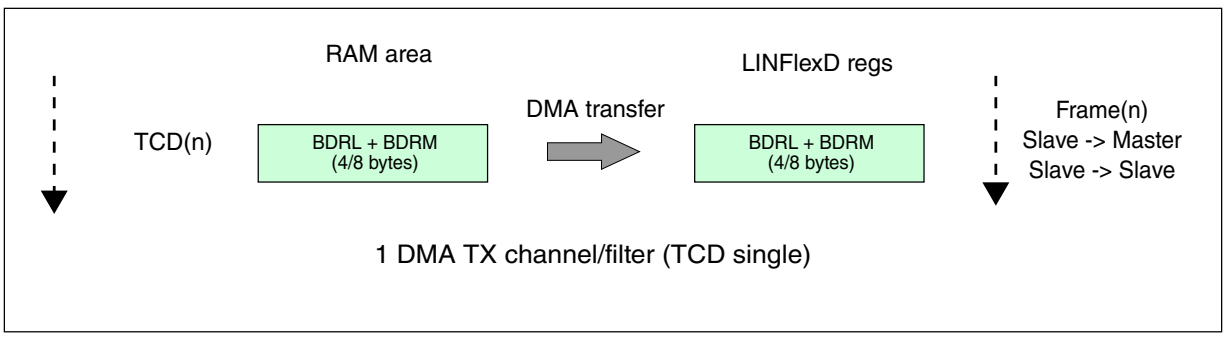


Figure 58-31. Slave node – TX memory map

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in Table 58-11.

Table 58-11. Slave node – Tx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1.The TCD setting (word transfer) is given in Table 58-12. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 58-12. TCD setting – Slave node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

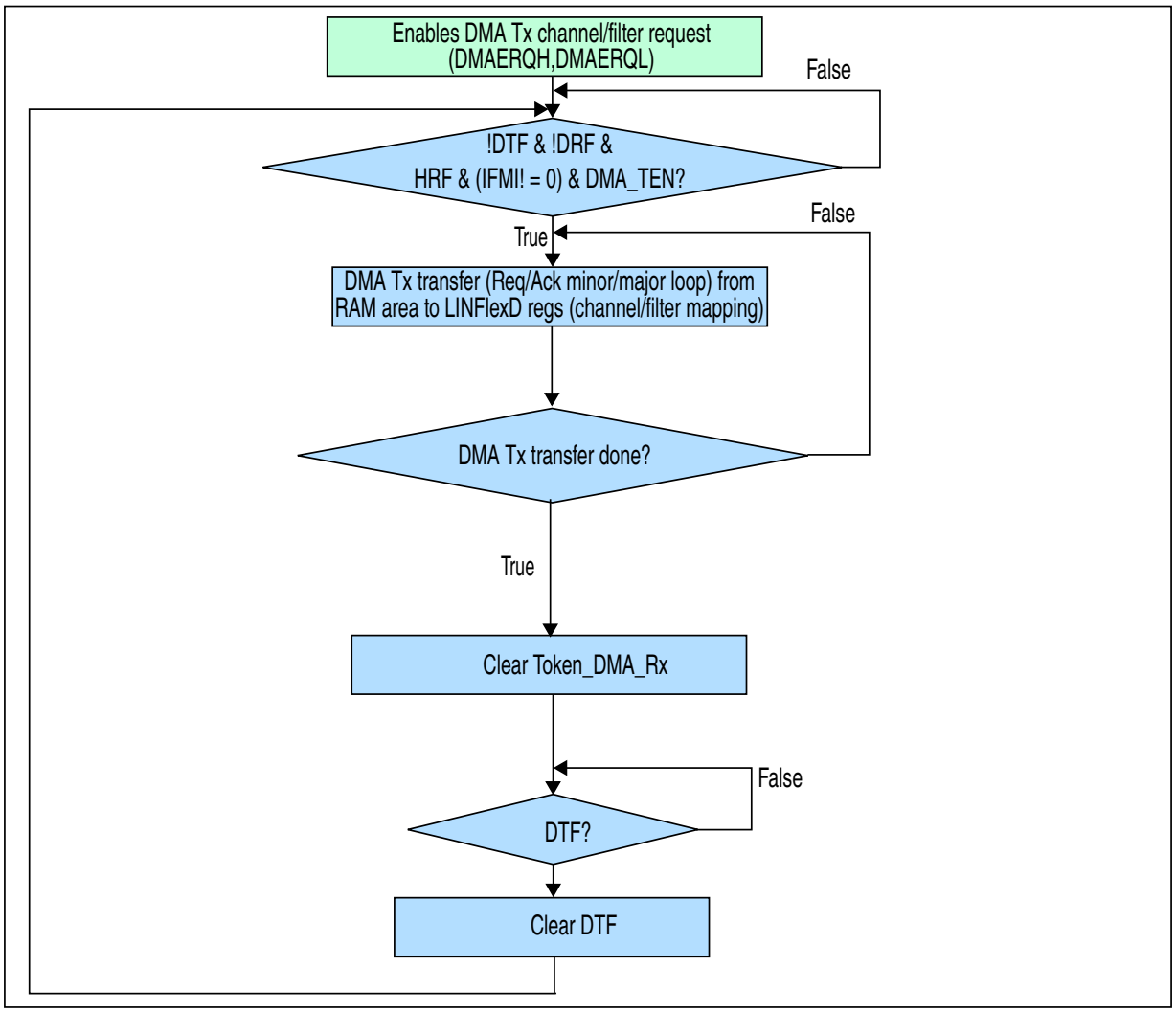


Figure 58-32. Slave node – DMA Tx FSM (concept scheme)

58.4.5.6 Slave node – RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

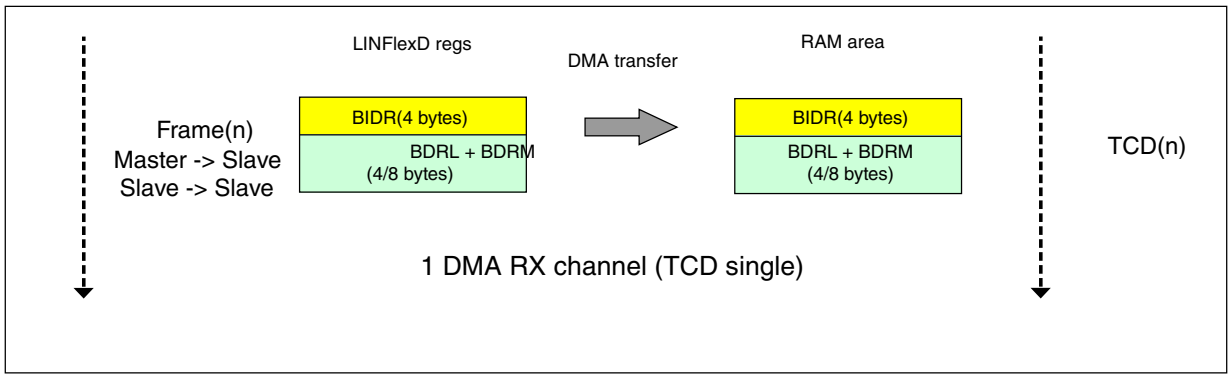


Figure 58-33. Slave node – RX memory map

The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, IFCR registers is given in Table 58-13.

Table 58-13. Slave node – Rx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in the following figure. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in Table 58-14. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 58-14. TCD setting – Slave node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—

Table continues on the next page...

Table 58-14. TCD setting – Slave node – Rx mode (continued)

TCD Field	Value	Description
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

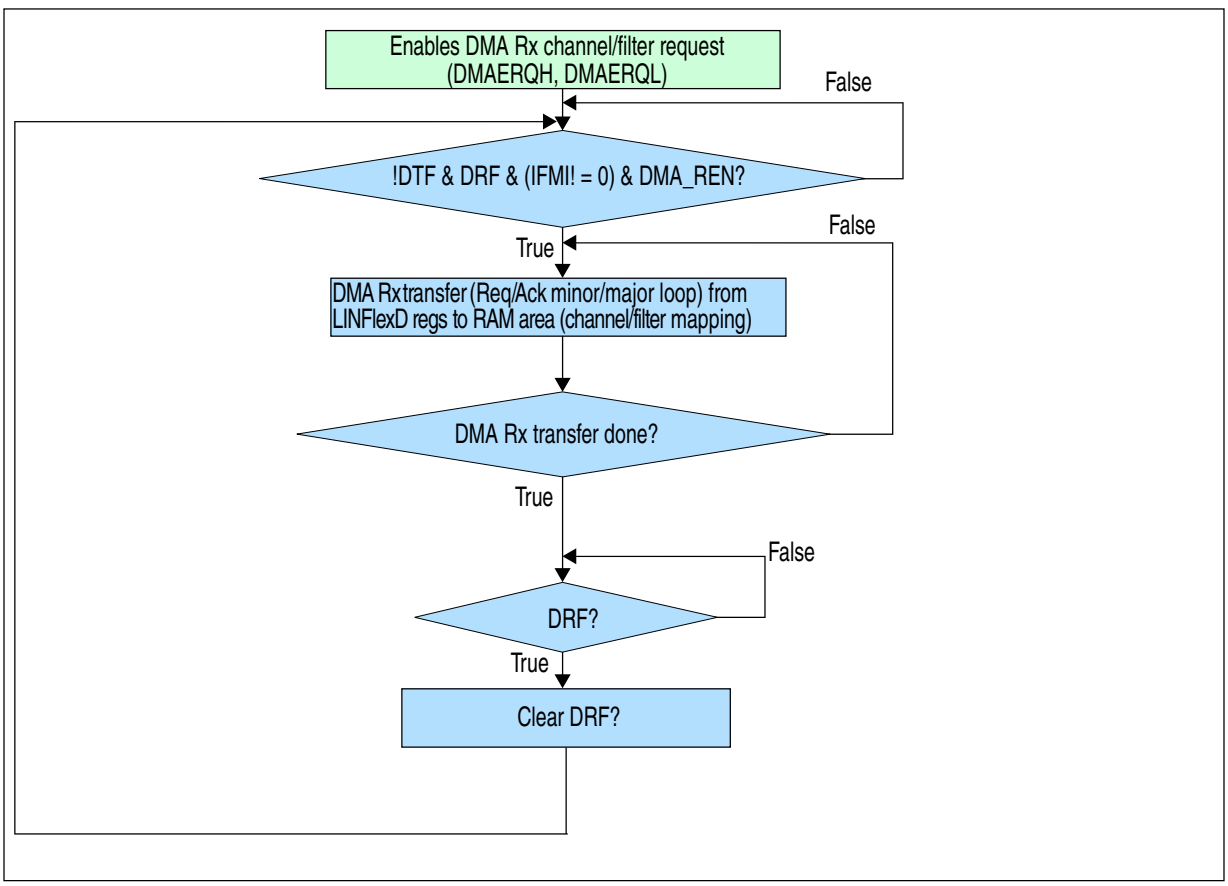


Figure 58-34. Slave node – DMA Rx FSM (concept scheme)

58.4.5.7 UART – TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Figure 58-35.

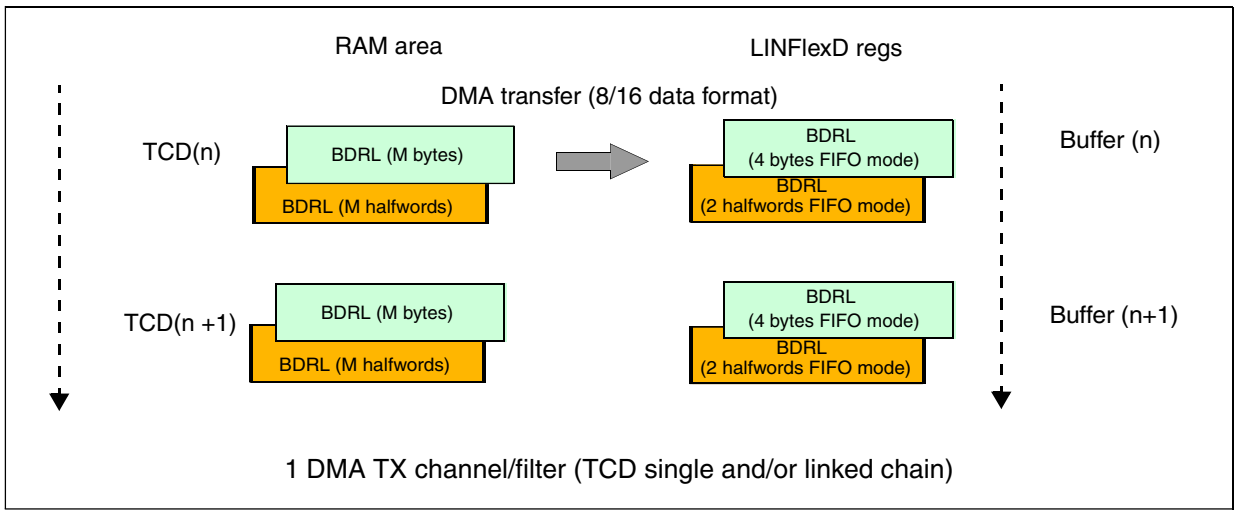


Figure 58-35. UART – TX memory map

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Absorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8 bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

The concept FSM to control the DMA Tx interface is given in Figure 58-36. DMA Tx FSM will move to Idle state if DMATXE[0] = 0.The TCD setting (typical case) is given in Table 58-15. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 58-15. TCD setting – UART – Tx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		RAM address

Table continues on the next page...

Table 58-15. TCD setting – UART – Tx mode (continued)

TCD Field	Value		Description
	8 bits data	16 bits data	
SOFF[15:0]	1	2	Byte/Half-word increment
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	–M	–M × 2	-
DADDR[31:0]	-		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

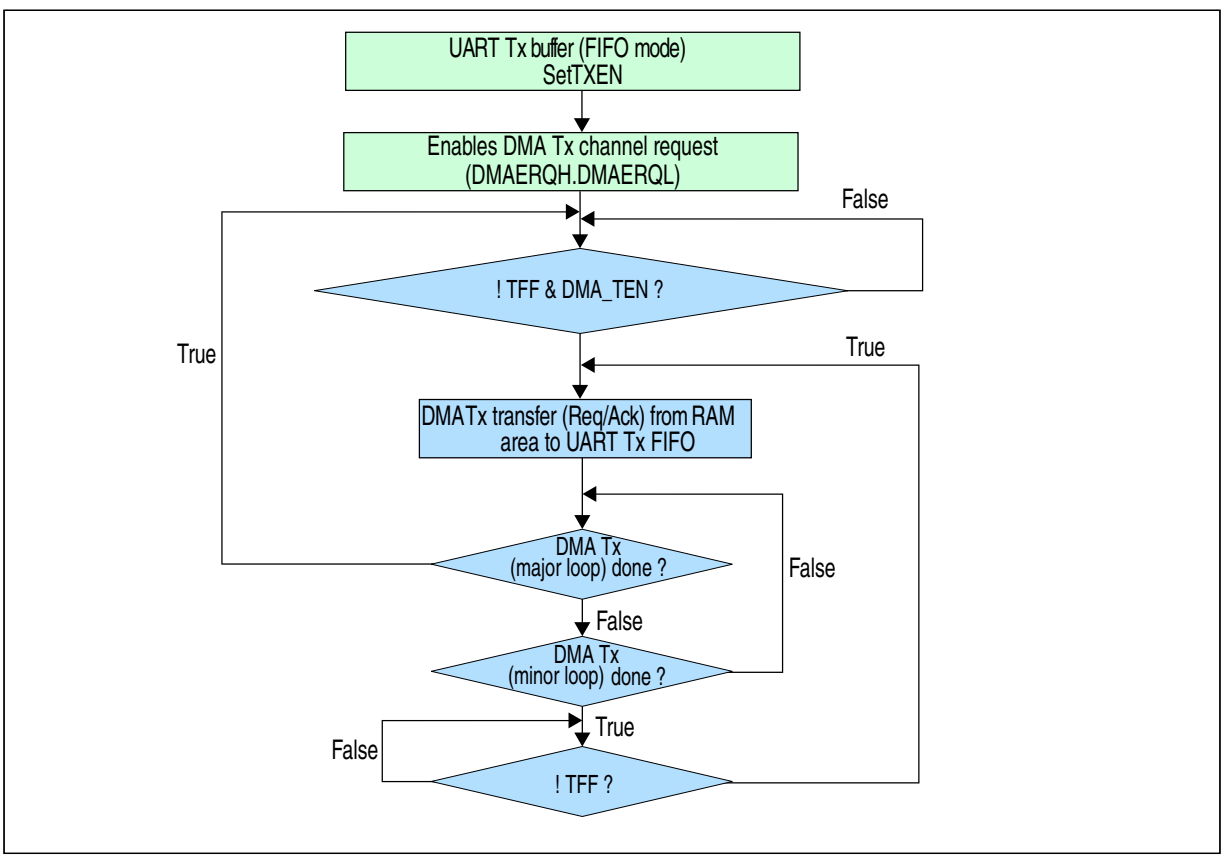


Figure 58-36. UART – DMA Tx FSM (concept scheme)

58.4.5.8 UART – RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Table 58-16.

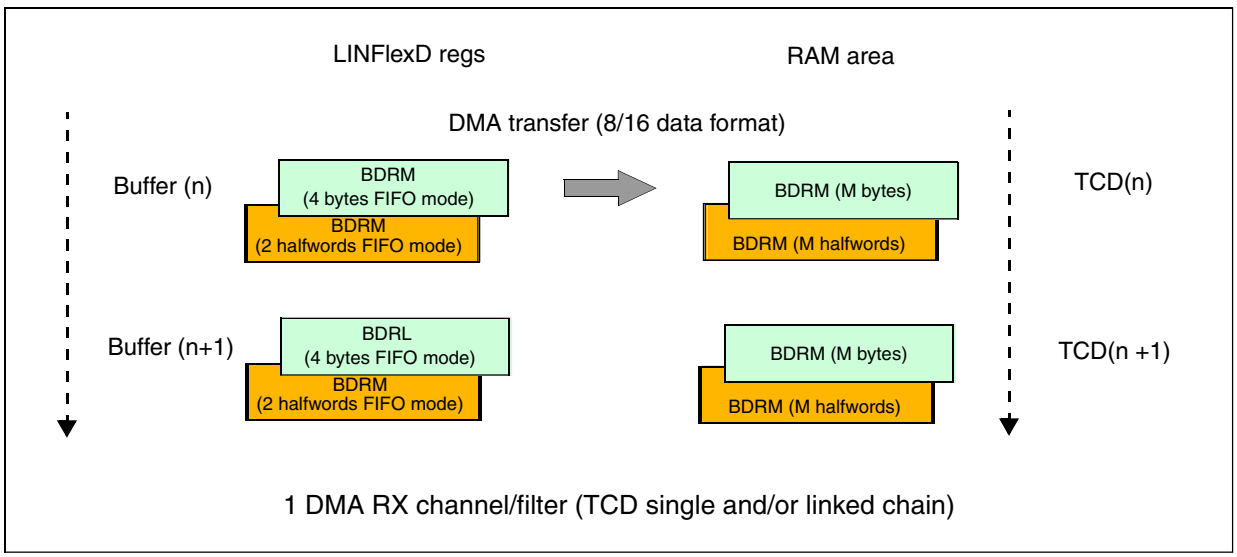


Figure 58-37. UART – RX memory map

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Absorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format

This will sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

The concept FSM to control the DMA Rx interface is given in Figure 58-38. DMA Rx FSM will move to Idle state if DMARXE[0] = 0.The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/ halfword as soon an entry is available in the Rx FIFO. A new software reset bit is

required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 58-16. TCD setting – UART – Rx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	0		
DADDR[31:0]	-		RAM address
DOFF[15:0]	1	2	Byte/Half-word increment
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

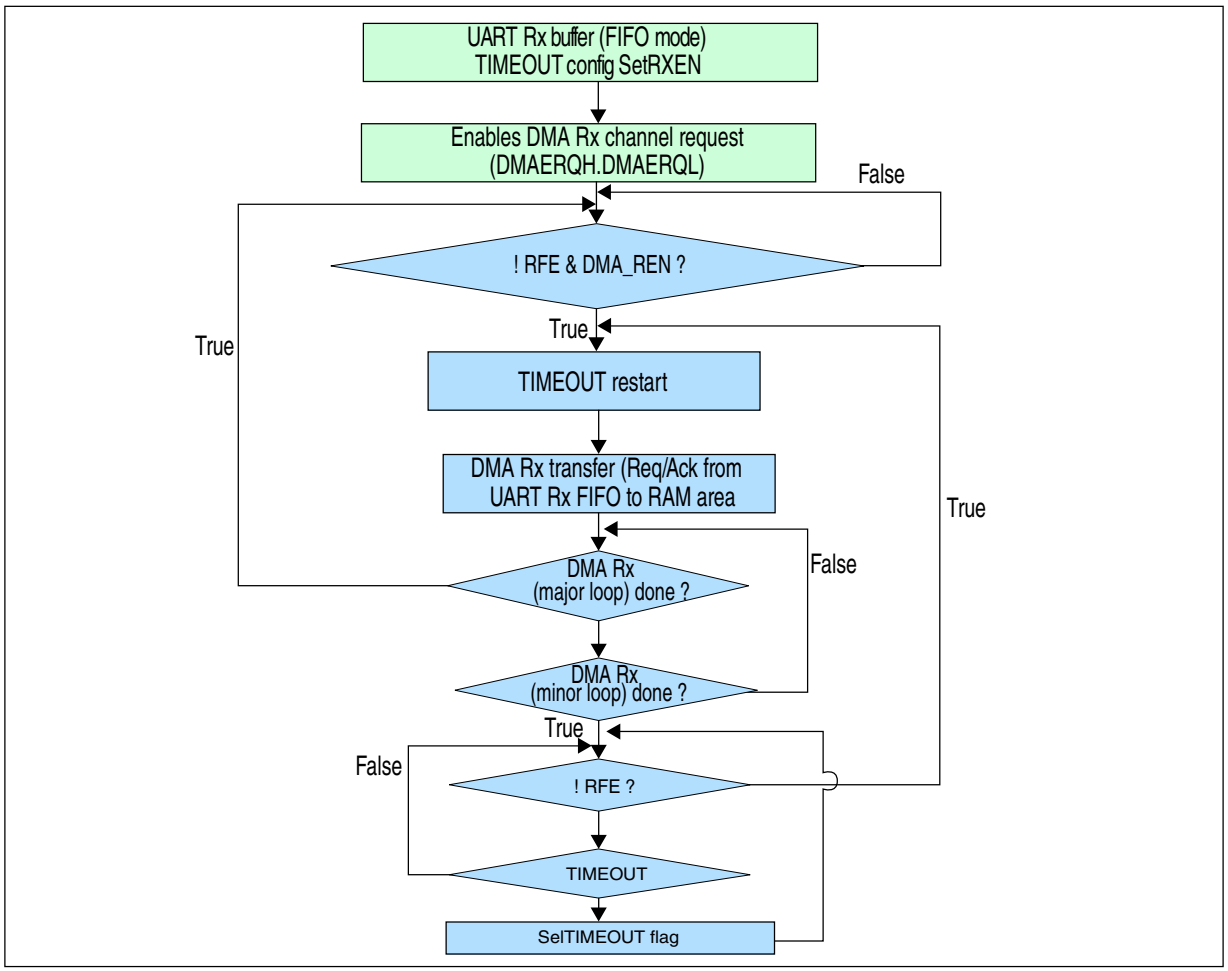


Figure 58-38. UART – DMA Rx FSM (concept scheme)

58.4.5.9 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index)
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses

- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

58.5 Memory map and register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of (generic) slave = 0 and (generic) slave = 1.

NOTE

In Master mode, the registers IFCR0-IFCR15 are not present and the corresponding absolute address of the below register changes as shown in the following table.

Table 58-17. Offsets of the register from offsets 8C to 9C

Register name	Master/slave mode	Master only mode
LINFlexD_GCR	0x8C	0x4C
LINFlexD_UARTPTO	0x90	0x50
LINFlexD_UARTCTO	0x94	0x54
LINFlexD_DMATXE	0x98	0x58
LINFlexD_DMARXE	0x9C	0x5C

NOTE

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in transfer error.

NOTE

Any access to A0 location will not generate a transfer error.

LINFlexD memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LIN Control Register 1 (LINFlexD_LINCR1)	32	R/W	See section	58.5.1/2790
4	LIN Interrupt enable register (LINFlexD_LINIER)	32	R/W	0000_0000h	58.5.2/2793
8	LIN Status Register (LINFlexD_LINSR)	32	R/W	See section	58.5.3/2795
C	LIN Error Status Register (LINFlexD_LINESR)	32	w1c	0000_0000h	58.5.4/2799
10	UART Mode Control Register (LINFlexD_UARTCR)	32	R/W	0000_0000h	58.5.5/2800
14	UART Mode Status Register (LINFlexD_UARTSR)	32	R/W	0000_0000h	58.5.6/2806
18	LIN Time-Out Control Status Register (LINFlexD_LINTCSR)	32	R/W	0000_0200h	58.5.7/2808
1C	LIN Output Compare Register (LINFlexD_LINOCR)	32	R/W	0000_FFFFh	58.5.8/2810
20	LIN Time-Out Control Register (LINFlexD_LINTOCR)	32	R/W	See section	58.5.9/2811
24	LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)	32	R/W	0000_0000h	58.5.10/ 2812
28	LIN Integer Baud Rate Register (LINFlexD_LINIBRR)	32	R/W	0000_0000h	58.5.11/ 2813
2C	LIN Checksum Field Register (LINFlexD_LINCFR)	32	R/W	0000_0000h	58.5.12/ 2813
30	LIN Control Register 2 (LINFlexD_LINCR2)	32	R/W	See section	58.5.13/ 2814
34	Buffer Identifier Register (LINFlexD_BIDR)	32	R/W	0000_0000h	58.5.14/ 2816
38	Buffer Data Register Least Significant (LINFlexD_BDRL)	32	R/W	0000_0000h	58.5.15/ 2817
3C	Buffer Data Register Most Significant (LINFlexD_BDRM)	32	R/W	0000_0000h	58.5.16/ 2818
40	Identifier Filter Enable Register (LINFlexD_IFER)	32	R/W	0000_0000h	58.5.17/ 2818
44	Identifier Filter Match Index (LINFlexD_IFMI)	32	R	0000_0000h	58.5.18/ 2819
48	Identifier Filter Mode Register (LINFlexD_IFMR)	32	R/W	0000_0000h	58.5.19/ 2819
4C	Identifier Filter Control Register (LINFlexD_IFCR0)	32	R/W	0000_0000h	58.5.20/ 2820
50	Identifier Filter Control Register (LINFlexD_IFCR1)	32	R/W	0000_0000h	58.5.20/ 2820
54	Identifier Filter Control Register (LINFlexD_IFCR2)	32	R/W	0000_0000h	58.5.20/ 2820
58	Identifier Filter Control Register (LINFlexD_IFCR3)	32	R/W	0000_0000h	58.5.20/ 2820
5C	Identifier Filter Control Register (LINFlexD_IFCR4)	32	R/W	0000_0000h	58.5.20/ 2820
60	Identifier Filter Control Register (LINFlexD_IFCR5)	32	R/W	0000_0000h	58.5.20/ 2820
64	Identifier Filter Control Register (LINFlexD_IFCR6)	32	R/W	0000_0000h	58.5.20/ 2820

Table continues on the next page...

LINFlexD memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68	Identifier Filter Control Register (LINFlexD_IFCR7)	32	R/W	0000_0000h	58.5.20/ 2820
6C	Identifier Filter Control Register (LINFlexD_IFCR8)	32	R/W	0000_0000h	58.5.20/ 2820
70	Identifier Filter Control Register (LINFlexD_IFCR9)	32	R/W	0000_0000h	58.5.20/ 2820
74	Identifier Filter Control Register (LINFlexD_IFCR10)	32	R/W	0000_0000h	58.5.20/ 2820
78	Identifier Filter Control Register (LINFlexD_IFCR11)	32	R/W	0000_0000h	58.5.20/ 2820
7C	Identifier Filter Control Register (LINFlexD_IFCR12)	32	R/W	0000_0000h	58.5.20/ 2820
80	Identifier Filter Control Register (LINFlexD_IFCR13)	32	R/W	0000_0000h	58.5.20/ 2820
84	Identifier Filter Control Register (LINFlexD_IFCR14)	32	R/W	0000_0000h	58.5.20/ 2820
88	Identifier Filter Control Register (LINFlexD_IFCR15)	32	R/W	0000_0000h	58.5.20/ 2820
8C	Global Control Register (LINFlexD_GCR)	32	R/W	0000_0000h	58.5.21/ 2821
90	UART Preset Timeout Register (LINFlexD_UARTPTO)	32	R/W	0000_0FFFh	58.5.22/ 2823
94	UART Current Timeout Register (LINFlexD_UARTCTO)	32	R	0000_0000h	58.5.23/ 2824
98	DMA Tx Enable Register (LINFlexD_DMATXE)	32	R/W	0000_0000h	58.5.24/ 2825
9C	DMA Rx Enable Register (LINFlexD_DMARXE)	32	R/W	0000_0000h	58.5.25/ 2825

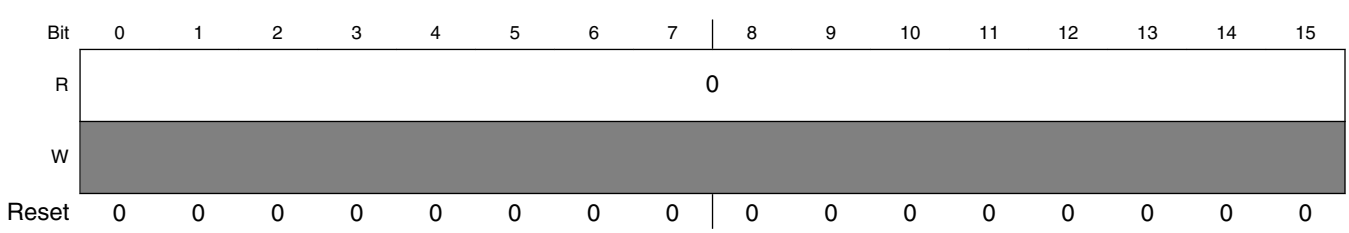
58.5.1 LIN Control Register 1 (LINFlexD_LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

NOTE

When accessing the LINCR1 register, each reserved bit should be written to its original reset value.

Address: 0h base + 0h offset = 0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CCD	CFD	LASE	AUTOWU	MBL				BF	Reserved	LBKM	MME	SSBL	RBLM	SLEEP	INIT
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

* Notes:
- MME field: If slave = 0, then MME bit is 1, else MME bit is 0

LINFlexD_LINCR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CCD	Checksum Calculation disable This bit can be written during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to sent a software calculated checksum/CRC (provided CFD is reset).
17 CFD	Checksum field disable This bit can be configured during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame
18 LASE	LIN Autosynchronization Enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0). Register bit can be read in any mode, written only in initialization mode. 0 Autosynchronization disabled 1 Autosynchronization enabled
19 AUTOWU	Auto Wakeup This bit can be configured during Initialization mode only. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set
20–23 MBL	Master Break Length These bits choose the length of the Sync break to be generated by the master. These bits can be programmed in Initialization mode only. They are read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0000 10-bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0001 11-bit break length 0010 12-bit break length 0011 13-bit break length 0100 14-bit break length 0101 15-bit break length 0110 16-bit break length 0111 17-bit break length 1000 18-bit break length 1001 19-bit break length 1010 20-bit break length 1011 21-bit break length 1100 22-bit break length 1101 23-bit break length 1110 36-bit (cooling) break length 1111 50-bit break length
24 BF	By-pass filter This bit can be programmed during Initialization mode only. **NOTE:** If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode. 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter
25 Reserved	This field is reserved. When accessing the LINCR1 register, each reserved bit should be written to its original reset value.
26 LBKM	Loop Back mode Refer to "Loop back mode" in Test mode . Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 MME	Master mode enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **NOTE:** If generic slave = 0, then this bit will read a '1' always, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode. 0 Slave Mode 1 Master Mode
28 SSBL	Slave Mode Sync Break Length This bit can be programmed in Initialization mode only. It is read only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 11 bit break length 1 10 bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
29 RBLM	Receiver Buffer Locked mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

58.5.2 LIN Interrupt enable register (LINFlexD_LINIER)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0		FEIE	BOIE	LSIE	WUIE	Reserved	TOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt enabled
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR. 0 No interrupt 1 Interrupt enabled
20 HEIE	Header Error Interrupt Enable An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **NOTE:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed. 0 No interrupt 1 Interrupt enabled
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR. 0 No interrupt 1 Interrupt enabled
25 LSIE	LIN state Interrupt enable Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag. 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum
26 WUIE	Wakeup interrupt enable If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated. 0 No interrupt 1 Interrupt enabled
27 Reserved	This field is reserved.

Table continues on the next page...

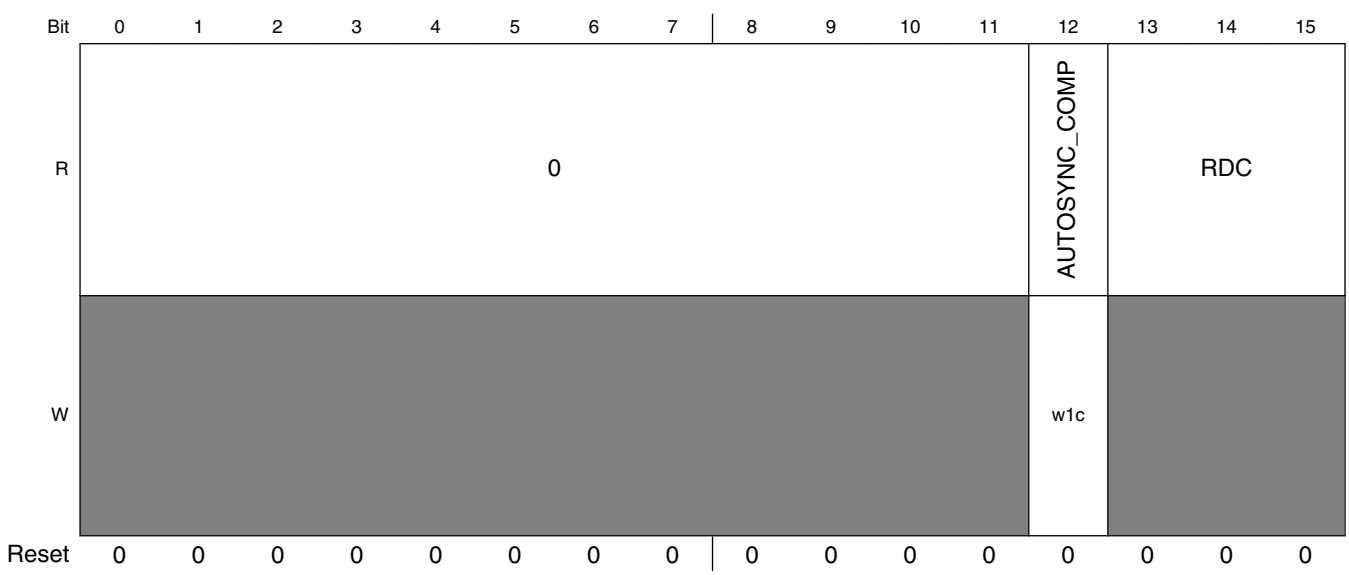
LINFlexD_LINIER field descriptions (continued)

Field	Description
28 TOIE	Timeout Interrupt Enable An interrupt is generated if this bit is set and UARTSR[TO] status bit is set (in UART mode). 0 No interrupt 1 Interrupt enabled
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
31 HRIE	Header Received Interrupt An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed. 0 No interrupt 1 Interrupt enabled

58.5.3 LIN Status Register (LINFlexD_LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0h base + 8h offset = 8h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0		RMB	DRBNE	RXbusy	RDI	WUF	0		DRF	DTF	HRF
W	1						w1c	w1c			w1c			w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	*	0	0	0	0	0	0

* Notes:
- RDI field: Reset value of RDI reflects the RX pin state

LINFlexD_LINSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 AUTOSYNC_ COMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode. **NOTE:** If autosynchronization is disabled, this bit is reserved.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
16–19 LINS	LIN state In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits. **NOTE:** The value of this bit field doesn't change by any write operation to it. Writing 0xF to this bit field clears the Rx interrupt, only when set due to the LIN state event. 0000 Sleep mode: LINFlexD is in Sleep mode, to save power consumption. 0001 Init mode: LINFlexD is in Initialization mode. 0010 Idle mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0011 Sync break: In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0100 Sync Del: In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0101 Sync Field: In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0110 Identifier Field: In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0111 Header Reception/Transmission: In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 1000 Data Reception/Data Transmission: In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 1001 Checksum: Data transmission/reception completed, checksum transmission/reception ongoing.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit. 0 Receiver is idle 1 Reception ongoing

Table continues on the next page...

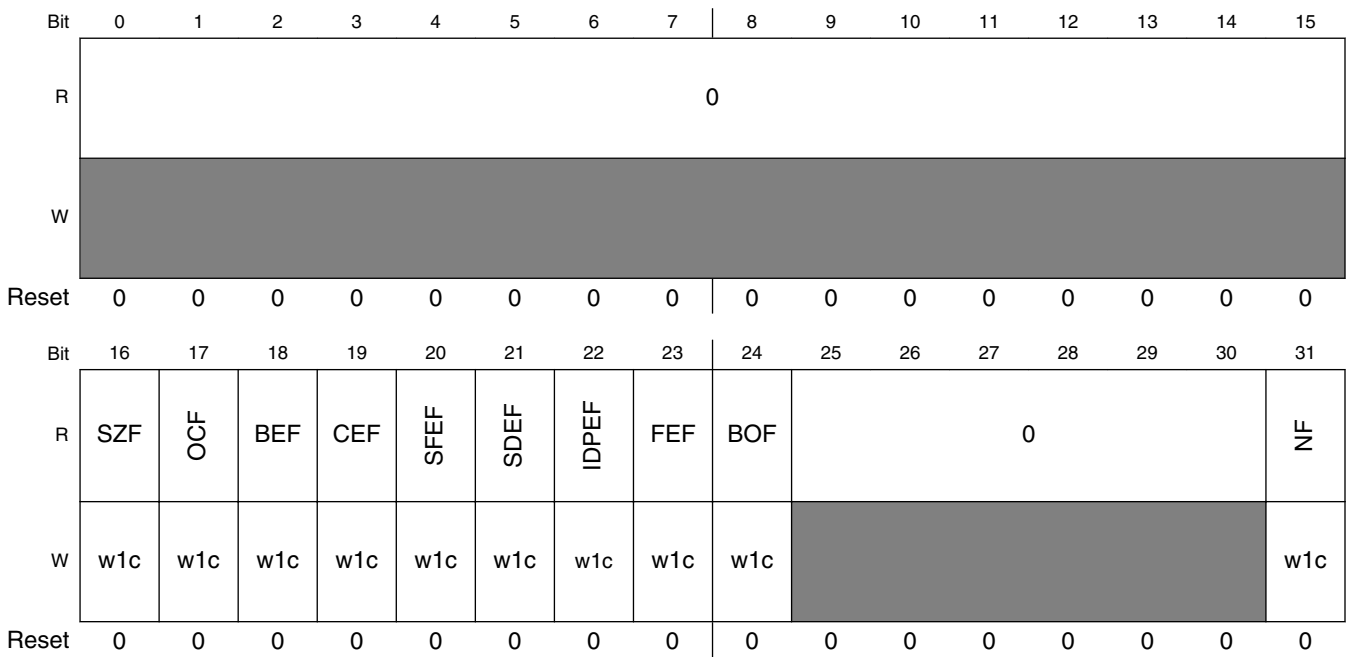
LINFlexD_LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **NOTE:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** For LIN mode, in case a bit error occurs (and IOBE is 1) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

58.5.4 LIN Error Status Register (LINFlexD_LINESR)

Refer to Errors for detailed description of all errors.

Address: 0h base + Ch offset = Ch



LINFlexD_LINESR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0: No output compare event occurred 1: In master mode, LINESR[OCF] flag is set when counter LINTCSR[CNT] has matched the content of LINOCR[OC2]. In slave mode, LINESR[OCF] is set when the content of the counter LINTCSR[CNT] has matched the content of LINOCR[OC1] or LINOCR[OC2]. This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.

Table continues on the next page...

LINFlexD_LINESR field descriptions (continued)

Field	Description
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **NOTE:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
25–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

58.5.5 UART Mode Control Register (LINFlexD_UARTCR)

NOTE

When accessing the UARTCR register, reserved bits should always be written to zero.

NOTE

- The LINFlexD module in UART mode does not support communication with Special Word Length (UARTCR[WLS] = 1) in buffer mode.
- The LINFlexD module in UART mode supports communication with Special Word Length (UARTCR[WLS] = 1) in FIFO mode only with UARTCR[WL1, WL0] = 1,1.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS	CSP			OSR				ROSE	NEF			DTU_PCETX	SBUR		WLS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL_TFC			RDFL_RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State Register bit can be read in any mode, written only in initialization mode. 0 UARTCTO monitors the number of bits to be received. 1 UARTCTO monitors the idle state of the reception line.
1–3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling. CSP can take the following range of values for a certain over sampling rate: **OSR / CSP** 4 / 2,3 5 / 2, 3, 4 6 / 3, 4, 5 8 / NA Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled. Allowed values are: 4, 5, 6 and 84 and 8 . Register bit can be read in any mode, written only in initialization mode.
8 ROSE	Reduced Over Sampling EnableRegister bit can be read in any mode, written only in initialization mode. 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate.
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received. Register bit can be read in any mode, written only in Initialization mode. Register bit is a read/write field.
12 DTU_PCETX	Disable Timeout in UART mode This bit can be programmed in Initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. **In Buffer mode:** This timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. **In FIFO mode:** When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. Notes: • Disable Timeout causes only the Timer to restart which enables Timeout again. • Timer reset means it resets and starts counting again and the Timeout is enabled. 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. This bit can be programmed in the initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 reserved
15 WLS	Special Word Length in UART mode

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	This bit can be programmed in initialization mode only when the UART bit is set. If this bit is set, setting WL and PCE bits has no effect in UART reception although parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode. Register bit can be read in any mode, written only in Initialization mode. 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support.
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 16 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes Register bit is a read/write field. TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 : empty 001 : 1 byte 010 : 2 bytes 011 : 3 bytes 100 : 4 bytes Others : reserved TDFLTFC can be programmed and are significative only when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. **NOTE:** When a debugger is connected, this counter will decrement if TxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
19–21 RDFL_RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 : Empty

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	001 : 1 byte 010 : 2 bytes/1.5 byte in case of WLS bit setting 011 : 3 bytes 100 : 4 bytes/2x1.5 byte in case of WLS bit setting Others - Reserved RDFLRFC can be programmed and are significative only when the UART bit is set. **NOTE:** In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint. **NOTE:** When a debugger is connected, this counter will decrement if RxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
22 RFBM	RFBM Rx Fifo/Buffer mode This bit can be programmed in Initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Rx Buffer mode enabled 1 Rx Fifo mode enabled (mandatory in DMA Rx mode)
23 TFBM	Tx Fifo/Buffer mode This bit can be programmed in initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Tx Buffer mode enabled 1 Tx Fifo mode enabled (mandatory in DMA Tx mode)
24 WL1	Word Length in UART mode This bit can be programmed in Initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in initialization mode.

WL1	WL0	Description
0	0	7 bits data + parity
0	1	8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1
1	0	15 bits data + parity
1	1	16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1

25 PC1	Parity Control This bit can be programmed in Initialization mode, only when UART bit is set. Register bit can be read in any mode, written only in initialization mode.

PC1	PC0	Description
0	0	Parity sent is Even

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>26
RxEn</td><td>Receiver Enable
This bit can be programmed only when the UART bit is set.
0 Receiver disabled
1 Receiver enabled</td></tr>
<tr><td>27
TxEn</td><td>Transmitter Enable
This bit can be programmed only when UART bit is set.
0 Transmitter disabled
1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed</td></tr>
<tr><td>28
PC0</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>29
PCE</td><td>Parity Control Enable
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Parity transmit/check Disable
1 Parity transmit/check Enable</td></tr>
<tr><td>30
WL0</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description		
	WL1	**WL0**	**Description**
	0	1	8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1
	1	0	15 bits data + parity
	1	1	16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1
31 UART	UART Mode This bit can be programmed in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 UART mode		

58.5.6 UART Mode Status Register (LINFlexD_UARTSR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRFRFE	DTFTFF	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. **NOTE:** For Baud rate above 1 Mbit/s, this flag is not usable. 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR
18–21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **NOTE:** Parity is checked only after complete frame is received. Following are the conditions when WL bits = 01 or 11 (either PE0 or PE2 is only set). • When PE0 is set, it indicates Parity error in either first or second byte received • When PE2 is set, it indicates Parity error in either third or fourth byte received 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	Release Message Buffer This bit should be cleared by software. This bit will reflect the same value as in LINSR, when in Initialization mode and the UART bit is set. 0 Buffer data is free 1 Buffer data ready to be read by software
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the new value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.

Table continues on the next page...

LINFlexD_UARTSR field descriptions (continued)

Field	Description
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.TOIE bit is set on the Error interrupt line in UART mode.
29 DRFRFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set when the configured number of valid stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error in case framing error is in the last STOP bit configured, parity error or overrun error. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTFTFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

58.5.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)

This register contains control and status bits for timeout feature.

NOTE

When the MODE bit is 0, any activity on the transmit or receive pins will cause an unwanted change in the value of the 8-bit field Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR). The LINFlexD module is enabled in LIN mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is changed from ‘1’ to ‘0’, then the old value of the Output Compare Value 1 (OC1) and Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR) is retained. As a consequence, if the module is reconfigured from UART to LIN mode, or LINTCSR MODE bit is changed from ‘1’ to ‘0’, an incorrect timeout exception is generated when a LIN communication starts. To avoid this, set mode bit to '1', then reconfigure the LINFlexD in LIN mode and then reset the MODE bit. Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR with 0xFFFF.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					MODE	IOT	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LINFlexD_LINTCSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MODE	Time-out counter mode This bit can be configured only during initialization. Register bit can be read in any mode, written only in initialization mode. **NOTE:** Always set LINTCSR[MODE] to 1 when UARTCR[UART] is 1. **NOTE:** Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR to 0xFFFF. 0 LIN mode 1 Output compare mode
22 IOT	Idle on timeout Register bit can be read in any mode, written only in initialization mode. This feature is applicable only when MODE bit in LINTCSR is cleared. 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event
23 TOCE	Time-out counter enable TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE. 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs.
24–31 CNT	Counter Value These bits reflect the value of a counter used for timeout. **NOTE:** For proper functionality of this counter, LINIBRR should be >= 5.

58.5.8 LIN Output Compare Register (LINFlexD_LINOCR)

This register contains the value to be compared to the LINTCSR:CNT value. This register is writable by software only in Output Compare Mode.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																OC2								OC1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_LINOCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 OC2	Output compare value 2
24–31 OC1	Output compare value 1

58.5.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	*	*	*	*	*	*	*

* Notes:

- HTO field: HTO reset values:

  HTO resets to 0011100b in Master only mode, if generic slave = 0.

  HTO resets to 0101100b in Master/Slave mode, if generic slave = 1.

LINFlexD_LINTOCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{Response\_Maximum} = 1.4 \times T_{Response\_Nominal}$
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. If Master mode is enabled then these bits will always reflect 28 (1.4 x 10 bits of sync + 1.4 x 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

58.5.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

NOTE

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												FBR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 FBR	Fractional Baud rates Register bit can be read in any mode, written only in initialization mode. 0000 Fraction(LDIV) = 0 0001 Fraction(LDIV) = 1/16 0010 Fraction(LDIV) = 2/16 0011 Fraction(LDIV) = 3/16 0100 Fraction(LDIV) = 4/16 0101 Fraction(LDIV) = 5/16 0110 Fraction(LDIV) = 6/16 0111 Fraction(LDIV) = 7/16 1000 Fraction(LDIV) = 8/16 1001 Fraction(LDIV) = 9/16 1010 Fraction(LDIV) = 10/16 1011 Fraction(LDIV) = 11/16 1100 Fraction(LDIV) = 12/16 1101 Fraction(LDIV) = 13/16 1110 Fraction(LDIV) = 14/16 1111 Fraction(LDIV) = 15/16

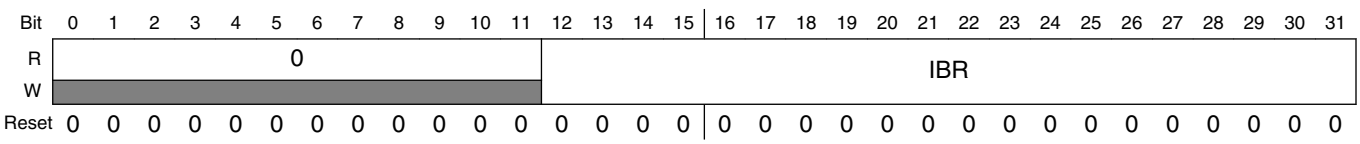
58.5.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 28h offset = 28h



LINFlexD_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. IBR = 0h: LIN clock disabled IBR = 1h: Mantissa (LDIV) = 1 ... IBR = FFFFEh: Mantissa (LDIV) = 1048574 IBR = FFFFFh: Mantissa (LDIV) = 1048575 Register bit can be read in any mode, written only in initialization mode.

58.5.12 LIN Checksum Field Register (LINFlexD_LINCFR)

NOTE

There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

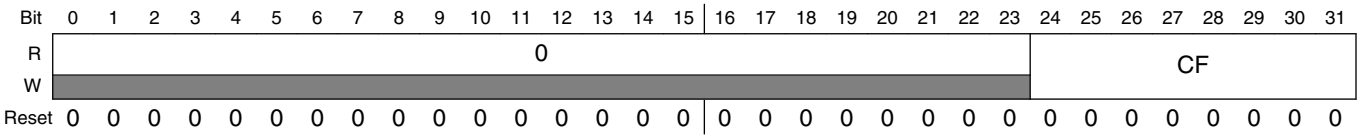
This register consists of checksum bits.

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None

Table continues on the next page...

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

Address: 0h base + 2Ch offset = 2Ch



LINFlexD_LINCFR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CF	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

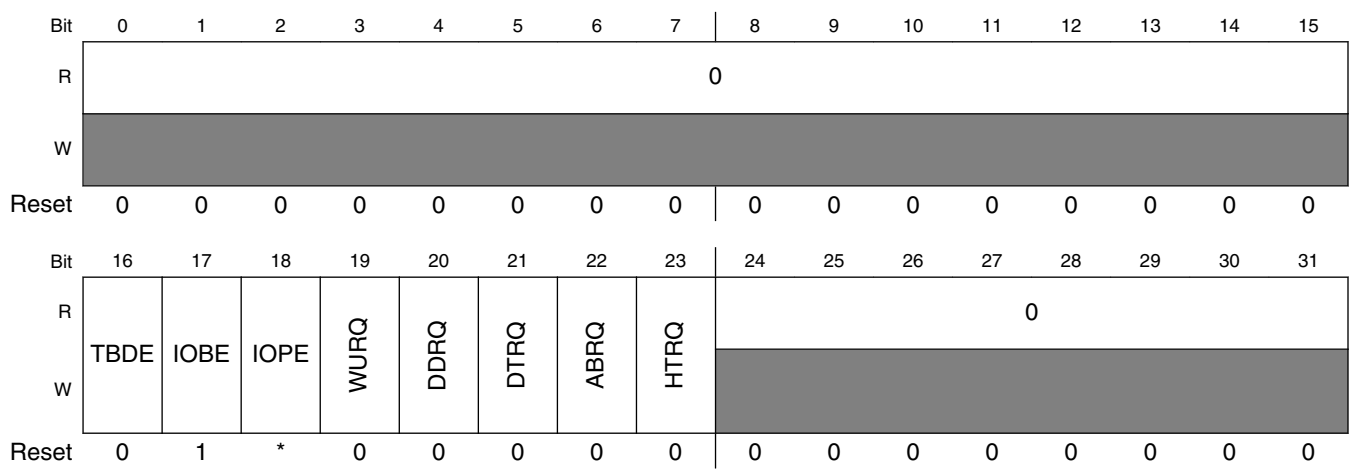
58.5.13 LIN Control Register 2 (LINFlexD_LINCR2)

This register includes control status bits related to buffer operations.

NOTE

When accessing the LINCR2 register, each reserved bit should be written to its original reset value.

Address: 0h base + 30h offset = 30h



* Notes:

- IOPE field: When slave = 0, this field always reads ‘0’ and cannot be programmed else this bit is programmable and reset value is 1.

LINFlexD_LINCR2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TBDE	Two Bit delimiter bit This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits
17 IOBE	Idle on Bit Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine
18 IOPE	Idle on Identifier Parity Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request. Register bit can be read/set by software
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered. Register bit can be read/set by software
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete. Register bit can be read/set by software
22 ABRQ	Abort Request Set by software to abort the current transmission.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also. Register bit can be read/set by software
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **NOTE:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.5.14 Buffer Identifier Register (LINFlexD_BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

NOTE

All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			DFL			DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BIDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–18 Reserved	This field is reserved.
19–21 DFL	Data Field Length Number of data bytes in the response part of the frame.

Table continues on the next page...

LINFlexD_BIDR field descriptions (continued)

Field	Description
	DFL = Number of data bytes - 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data filed only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

58.5.15 Buffer Data Register Least Significant (LINFlexD_BDRL)

This register is a part of an 8-byte data buffer.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRL field descriptions

Field	Description
0–7 DATA3	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2	Data Byte 2 Data byte 2 of the data field.
16–23 DATA1	Data Byte 1 Data byte 1of the data field.
24–31 DATA0	Data Byte 0 Data byte 0 of the data field.

58.5.16 Buffer Data Register Most Significant (LINFlexD_BDRM)

This register is a part of an 8-byte data buffer.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA7								DATA6								DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRM field descriptions

Field	Description
0–7 DATA7	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6	Data Byte 6 Data byte 6 of the data field.
16–23 DATA5	Data Byte 5 Data byte 5of the data field.
24–31 DATA4	Data Byte 4 Data byte 4 of the data field.

58.5.17 Identifier Filter Enable Register (LINFlexD_IFER)

This register enables/disables a particular filter.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																FACT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

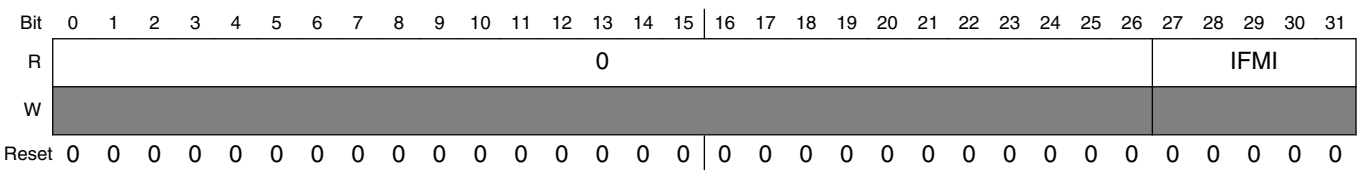
LINFlexD_IFER field descriptions (continued)

Field	Description
	1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **NOTE:** Refer to chip configuration details to see the number of filters used in the device.

58.5.18 Identifier Filter Match Index (LINFlexD_IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0h base + 44h offset = 44h



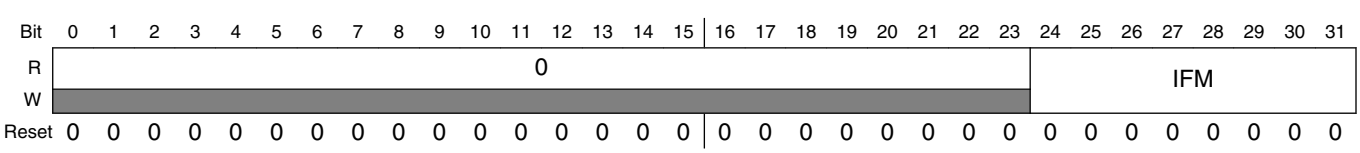
LINFlexD_IFMI field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Slave mode for more details).

58.5.19 Identifier Filter Mode Register (LINFlexD_IFMR)

This register configures the modes of filters.

Address: 0h base + 48h offset = 48h



LINFlexD_IFMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 IFM	Filter mode Register bit can be read in any mode, written only in initialization mode. 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

58.5.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode.

Even-numbered instances (IFCR0, IFCR2, IFCR4, ...) of this register act as filters in both Identifier list and Identifier mask modes.

Odd-numbered instances (IFCR1, IFCR3, IFCR5, ...) of this register act:

- As filters in Identifier list mode
- As a mask for the preceding-numbered register in identifier mask mode

Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			DFL			DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–18 Reserved	This field is reserved.
19–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL = Number of data bytes - 1

Table continues on the next page...

LINFlexD_IFCR*n* field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in initialization mode.
22 DIR	Direction This bit controls the direction of the data field. Register bit can be read in any mode, written only in initialization mode. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. Register bit can be read in any mode, written only in initialization mode. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. Register bit can be read in any mode, written only in initialization mode.

58.5.21 Global Control Register (LINFlexD_GCR)

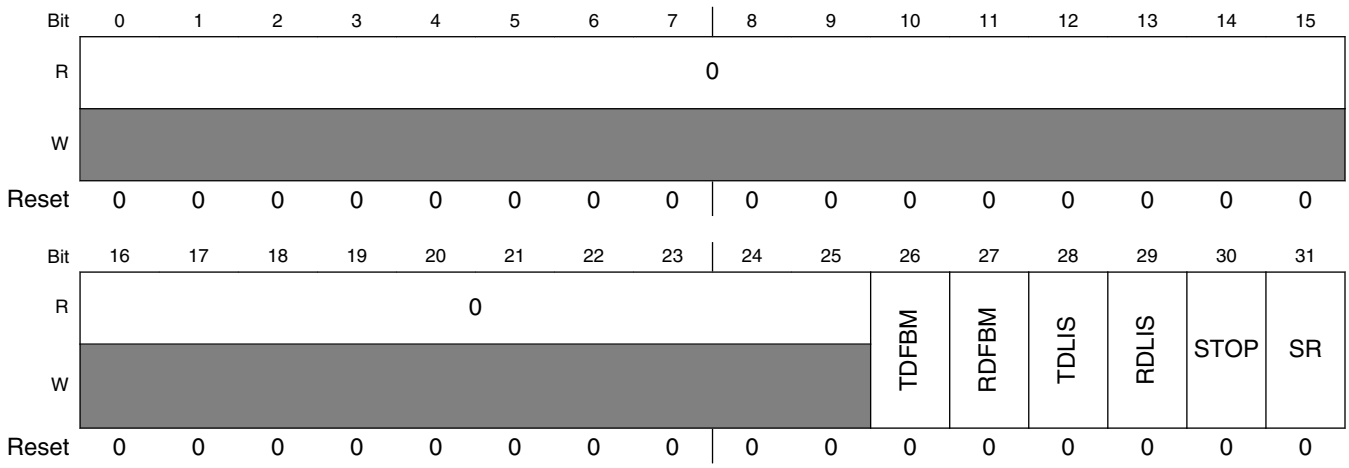
This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

The address offset depends on the no_of_filters. Refer to the chip configuration details for the number of filters used in this device.

Table 58-18. Register fields reset by SR

Register	Comment
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset
LINESR	All fields are reset
LINTCSR	Only CNT[0:7] is reset
UARTSR	All fields except RFE & TFF are reset
UARTCR	Only TFC & RFC are reset
UARTCTO	All fields are reset

Address: 0h base + 8Ch offset = 8Ch



LINFlexD_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Received data is not inverted 1 Received data is inverted

Table continues on the next page...

LINFlexD_GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). Register bit can be read in any mode, written only in initialization mode. 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). Register bit can be written only be software in initialization mode. Bit is always read 0 by software. Table 58-18 describes the register fields reset by SR.

58.5.22 UART Preset Timeout Register (LINFlexD_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

The address offset depends on no_of_filters. Refer to the chip configuration details to see the number of filters used in the device.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																PTO															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

58.5.23 UART Current Timeout Register (LINFlexD_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see UART Preset Timeout Register (LINFlexD_UARTPTO) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

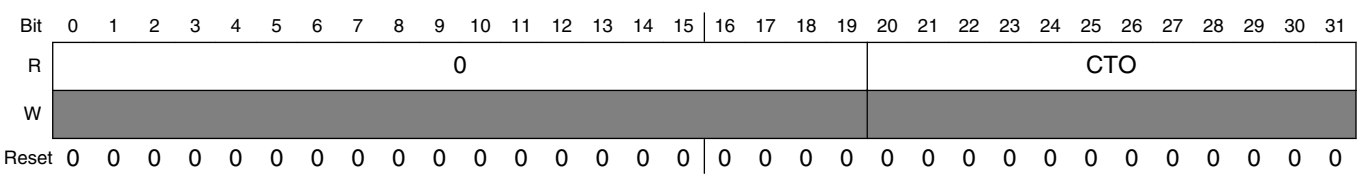
- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 94h offset = 94h



LINFlexD_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout

Table continues on the next page...

LINFlexD_UARTCTO field descriptions (continued)

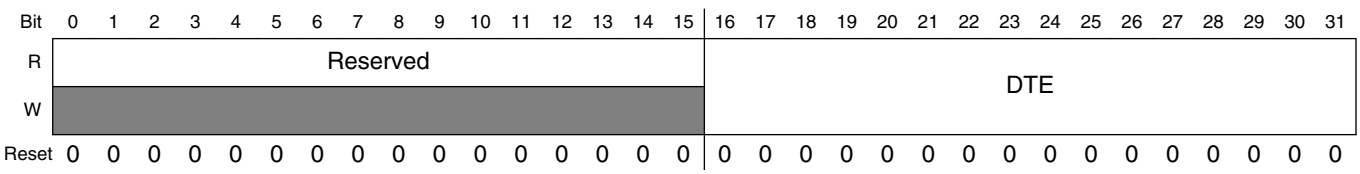
Field	Description
	CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

58.5.24 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h



LINFlexD_DMATXE field descriptions

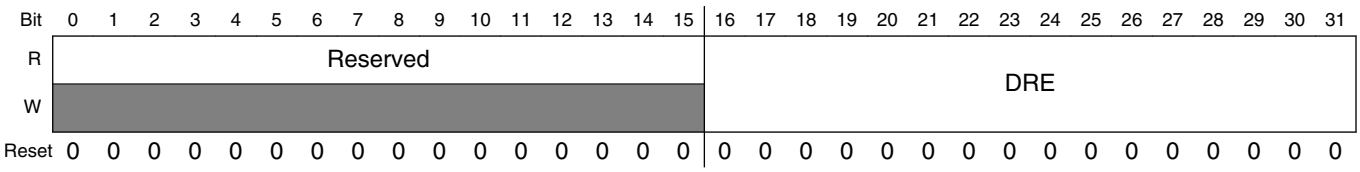
Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{TX\_CH\_NUM}$. These lies from 0 to (31 - $2^{TX\_CH\_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX\_CH\_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX\_CH\_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

58.5.25 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch



LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{RX\_CH\_NUM}$. These lies from 0 to (31 - $2^{RX\_CH\_NUM}$). Refer to the chip configuration details for the value of RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable **NOTE:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. The size is [$2^{RX\_CH\_NUM}$ -1 : 0]. The position of these bits are (32 - $2^{RX\_CH\_NUM}$) to 31. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

58.6 Programming considerations

The next subsections describe the various configurations in which the LINFlexD module can be used.

58.6.1 Master node

LINFlexD acts as Master when the MME bit is set.

58.6.1.1 Transmitter

Transmitter Master Node - Transmitter configuration.

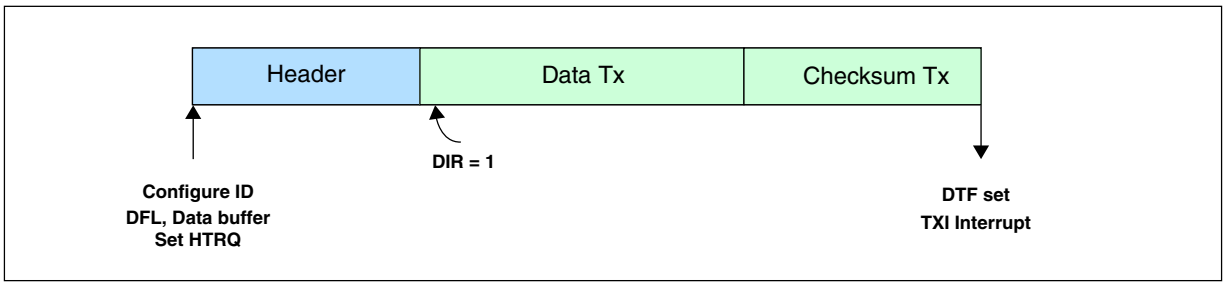


Figure 58-39. Master node – Transmitter configuration

58.6.1.2 Receiver

Receiver Master Mode - RX Configuration

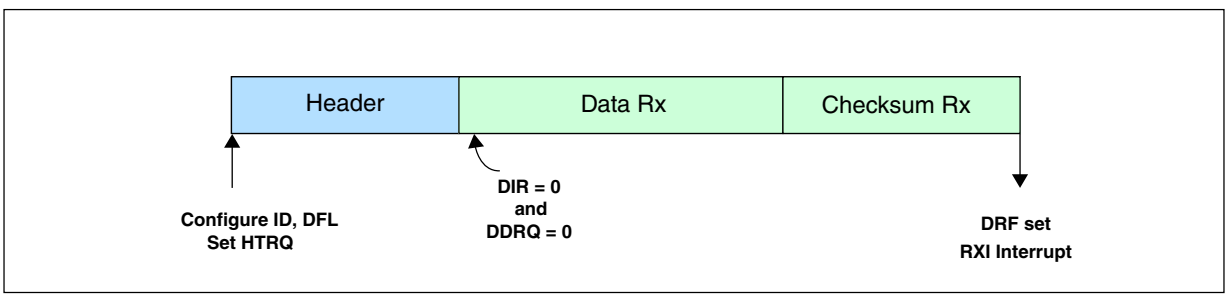


Figure 58-40. Master node – Receiver configuration

58.6.1.3 Transmitter, Bit Error

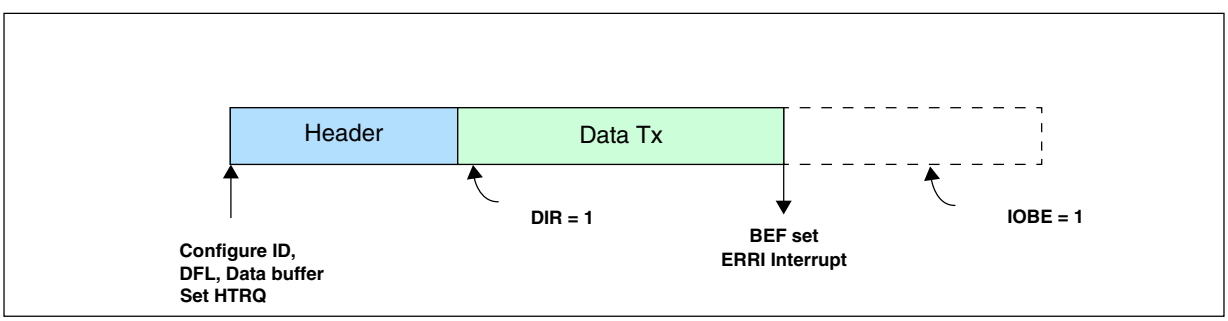


Figure 58-41. Master node – Transmitter, Bit Error configuration

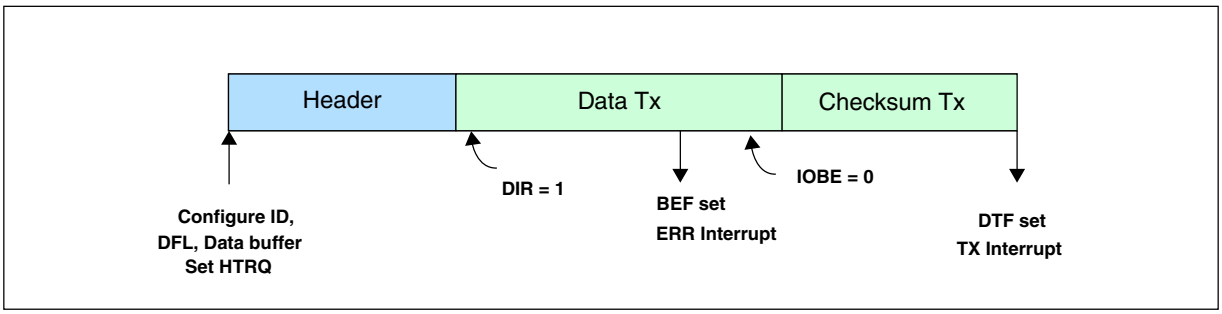


Figure 58-42. Master node – Transmitter, Checksum Error configuration

58.6.1.4 Receiver, Checksum Error

Checksum Error for Receiver.

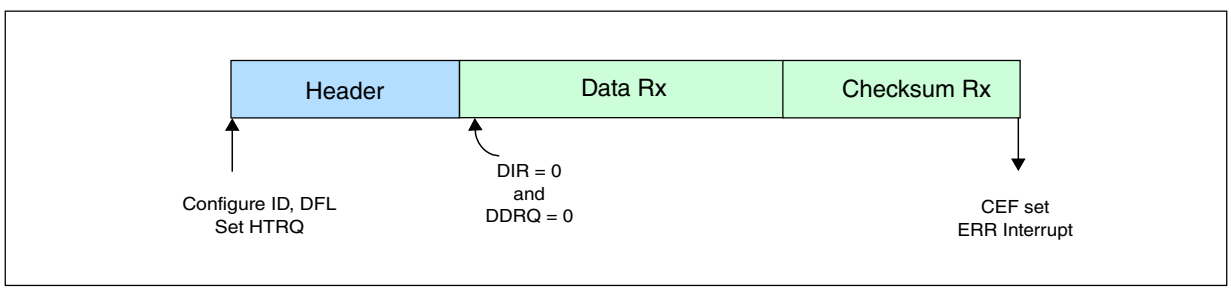


Figure 58-43. Master node – Receiver, Checksum Error configuration

58.6.2 Slave node

Slave node (transmitter).

58.6.2.1 Transmitter (no identifier filters)

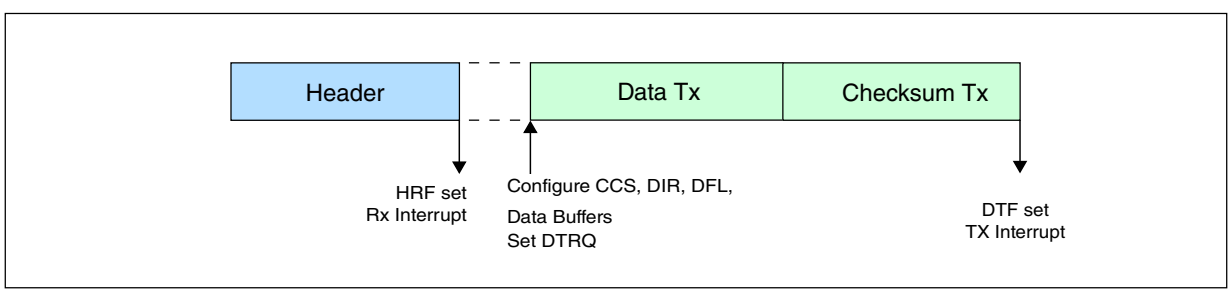


Figure 58-44. Slave node – Transmitter (no identifier filters) configuration

58.6.2.2 Receiver (no identifier filters)

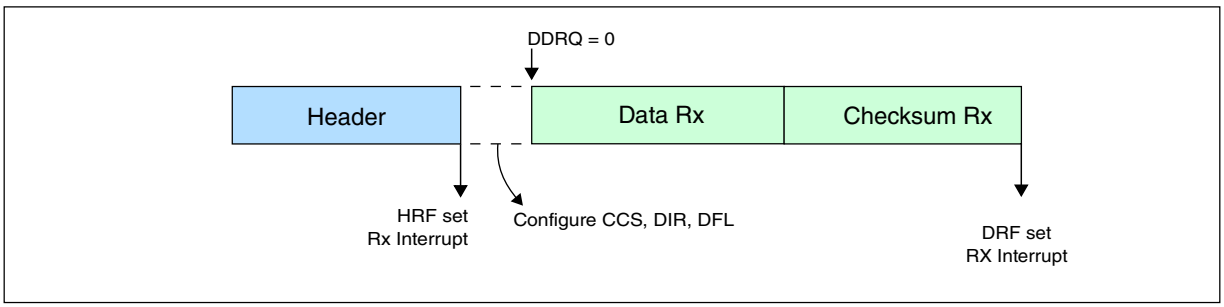


Figure 58-45. Slave node – Receiver (no identifier filters) configuration

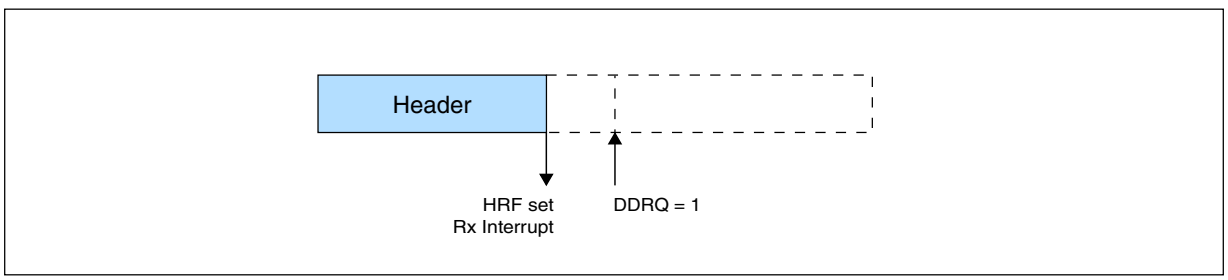


Figure 58-46. Slave node – Receiver (no identifier filters) configuration

58.6.2.3 No filters, Transmitter, Bit error

The following figure shows Slave node - No filters, Transmitter, Bit error configuration.

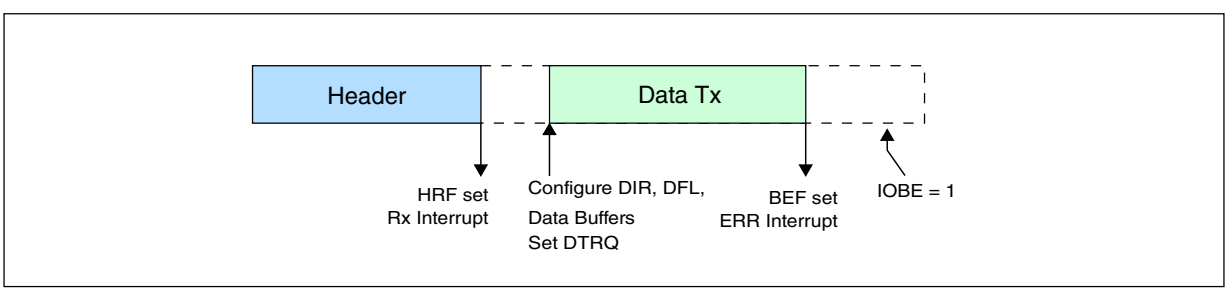


Figure 58-47. Slave node – No filters, Transmitter, Bit error configuration

58.6.2.4 No filters, Receiver, Checksum Error

The following figure shows Slave node - No filters, Receiver, Checksum error configuration.

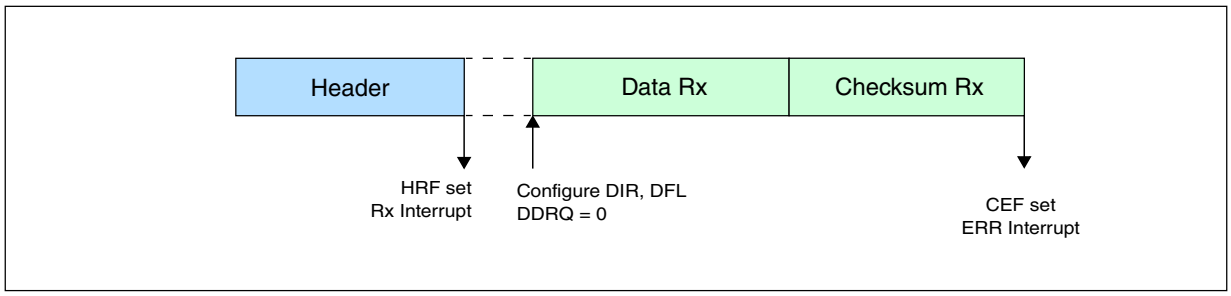


Figure 58-48. Slave node – No filters, Receiver, Checksum error configuration

58.6.2.5 At least one TX filter, BF is reset, ID matches filter

This configuration can be used in case slave never receives data, for example, sensor.

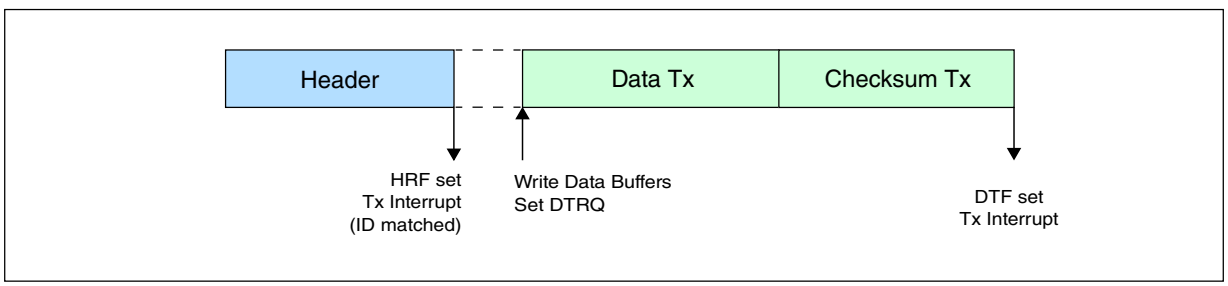


Figure 58-49. Slave node – one TX filter configuration

58.6.2.6 At least one RX filter, BF reset, ID matches filter

The following figure shows Slave node - one RX filter configuration.

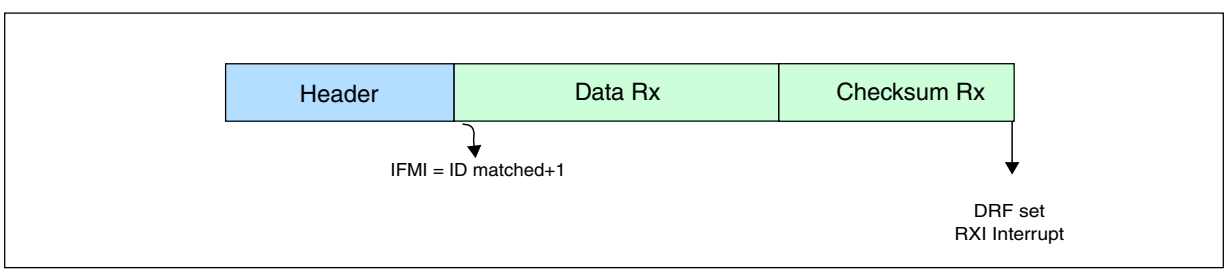


Figure 58-50. Slave node – one RX filter configuration

58.6.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

The following figure shows RX only, TX only, RX and TX filters configuration.

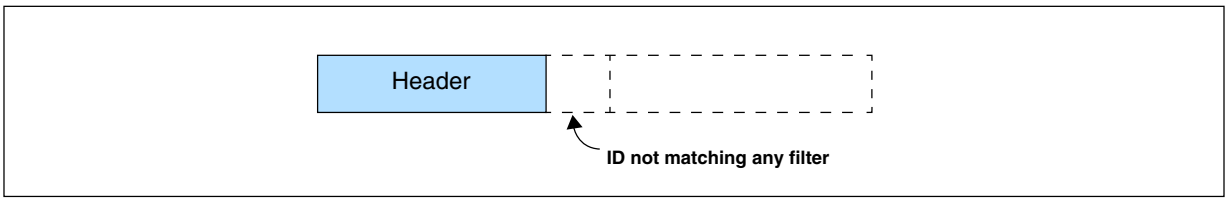


Figure 58-51. RX only, TX only, RX & TX filters configuration

58.6.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

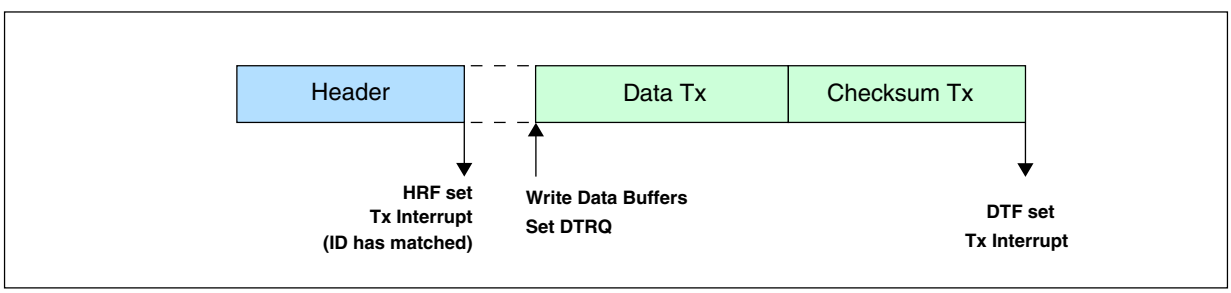


Figure 58-52. TX filter, BF is set configuration – ID has matched

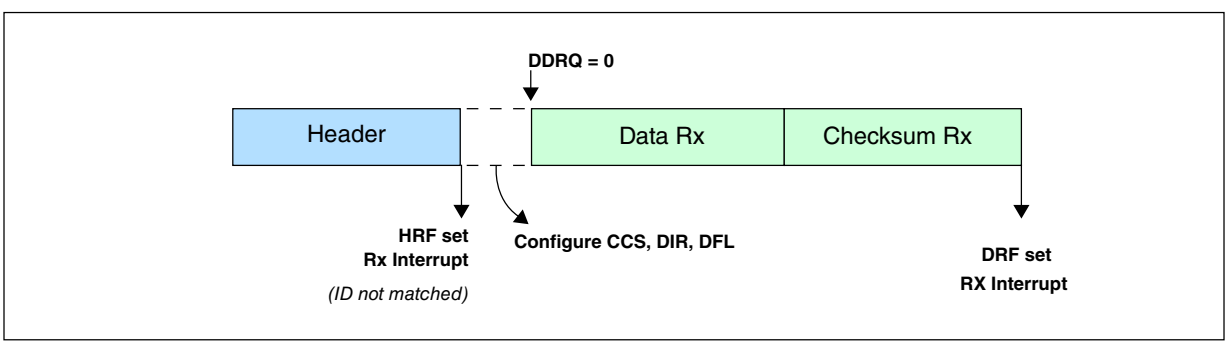


Figure 58-53. TX filter, BF is set configuration – ID not matched

58.6.2.9 RX filter, BF is set

The following figures show RX filter configurations.

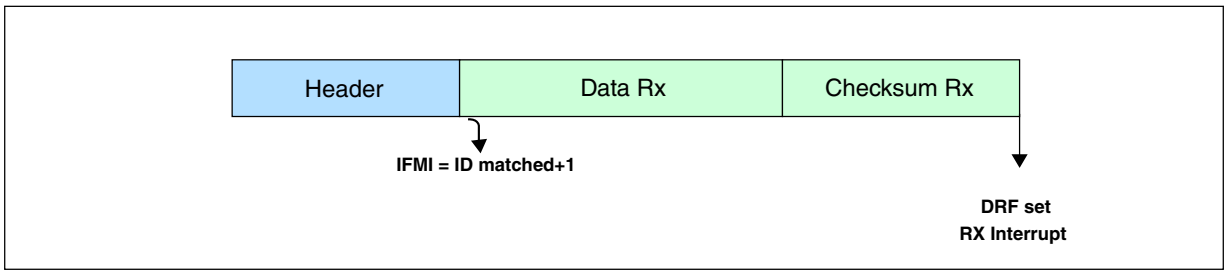


Figure 58-54. RX filter, BF is set configuration – ID matched

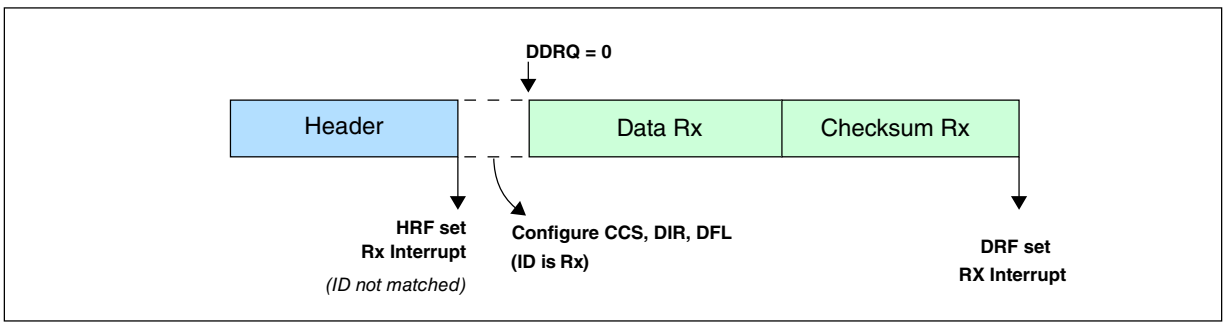


Figure 58-55. RX filter, BF is set configuration – ID not matched (ID is Rx)

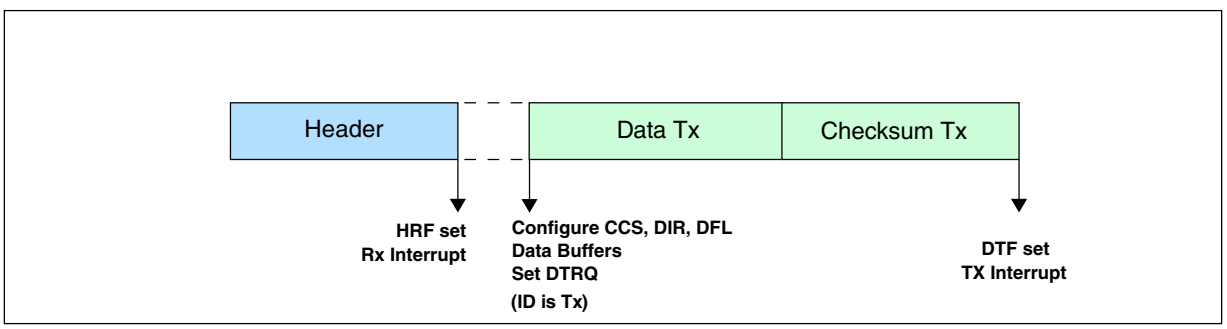


Figure 58-56. RX filter, BF is set configuration – (ID is Tx)

58.6.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

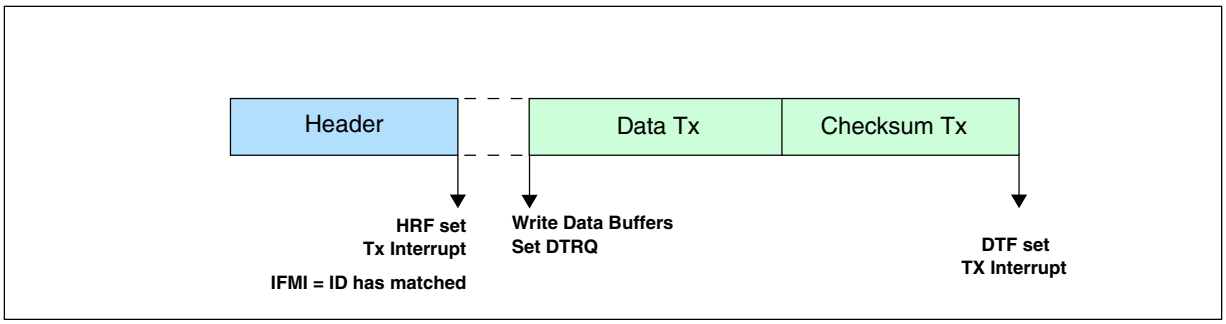


Figure 58-57. TX filter, RX filter, BF set configuration – (ID matched)

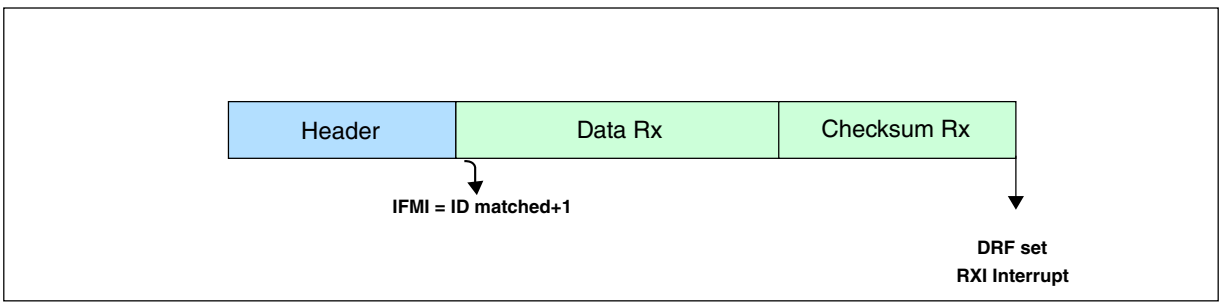


Figure 58-58. TX filter, RX filter, BF set configuration – (ID matched + 1)

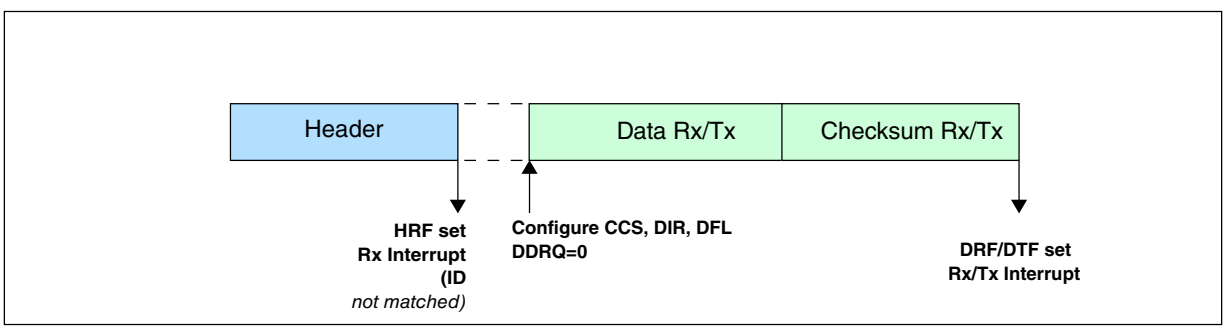


Figure 58-59. TX filter, RX filter, BF set configuration – (ID not matched, RX/TX Interrupt)

58.6.3 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

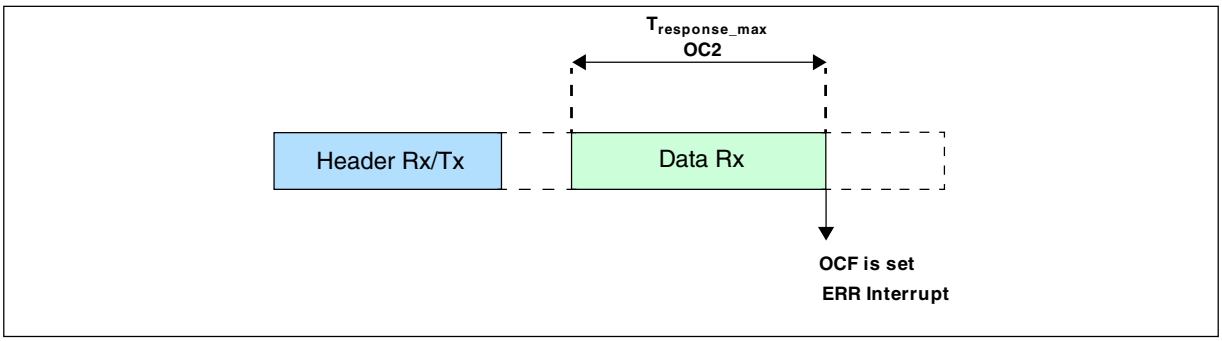


Figure 58-60. Response timeout

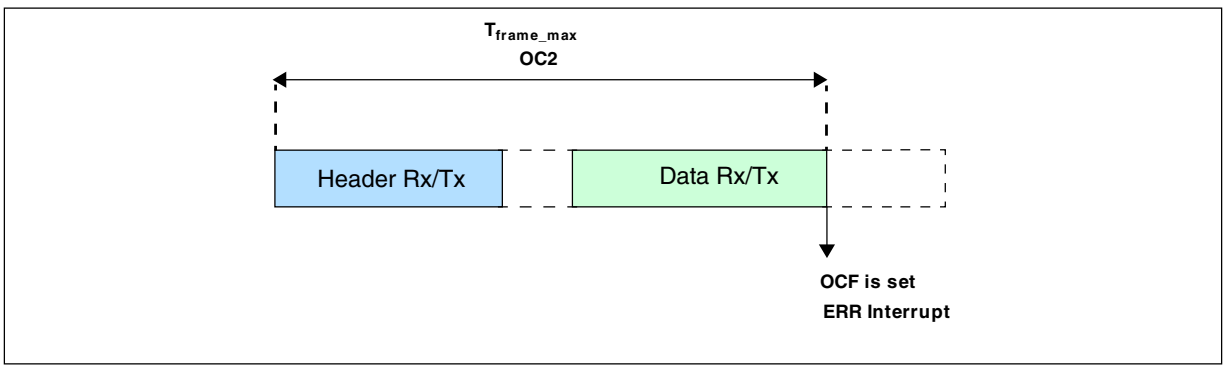


Figure 58-61. Frame timeout

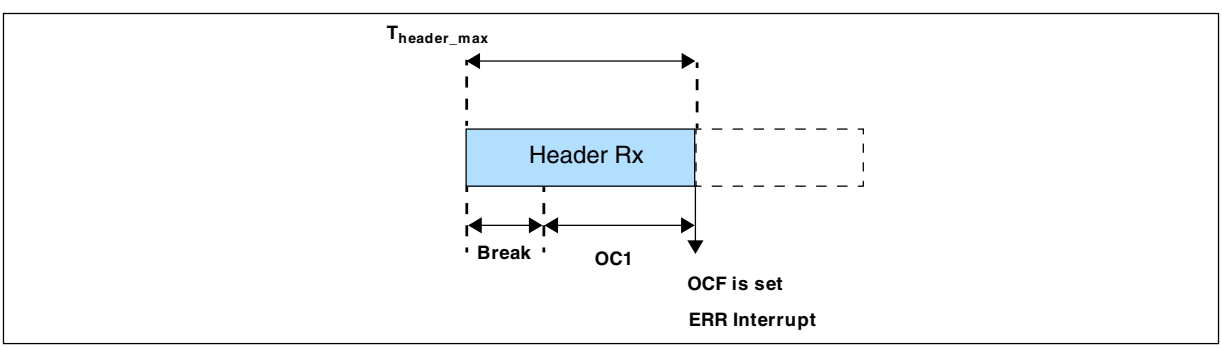


Figure 58-62. Header timeout

58.6.4 UART mode

The following figure shows UART mode configuration.

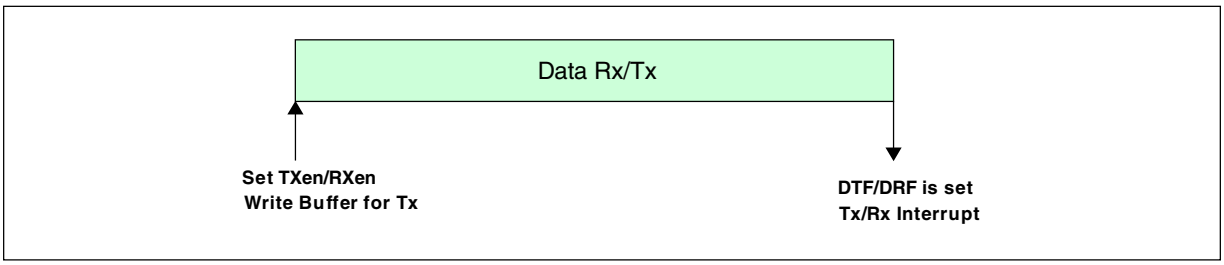


Figure 58-63. UART mode configuration

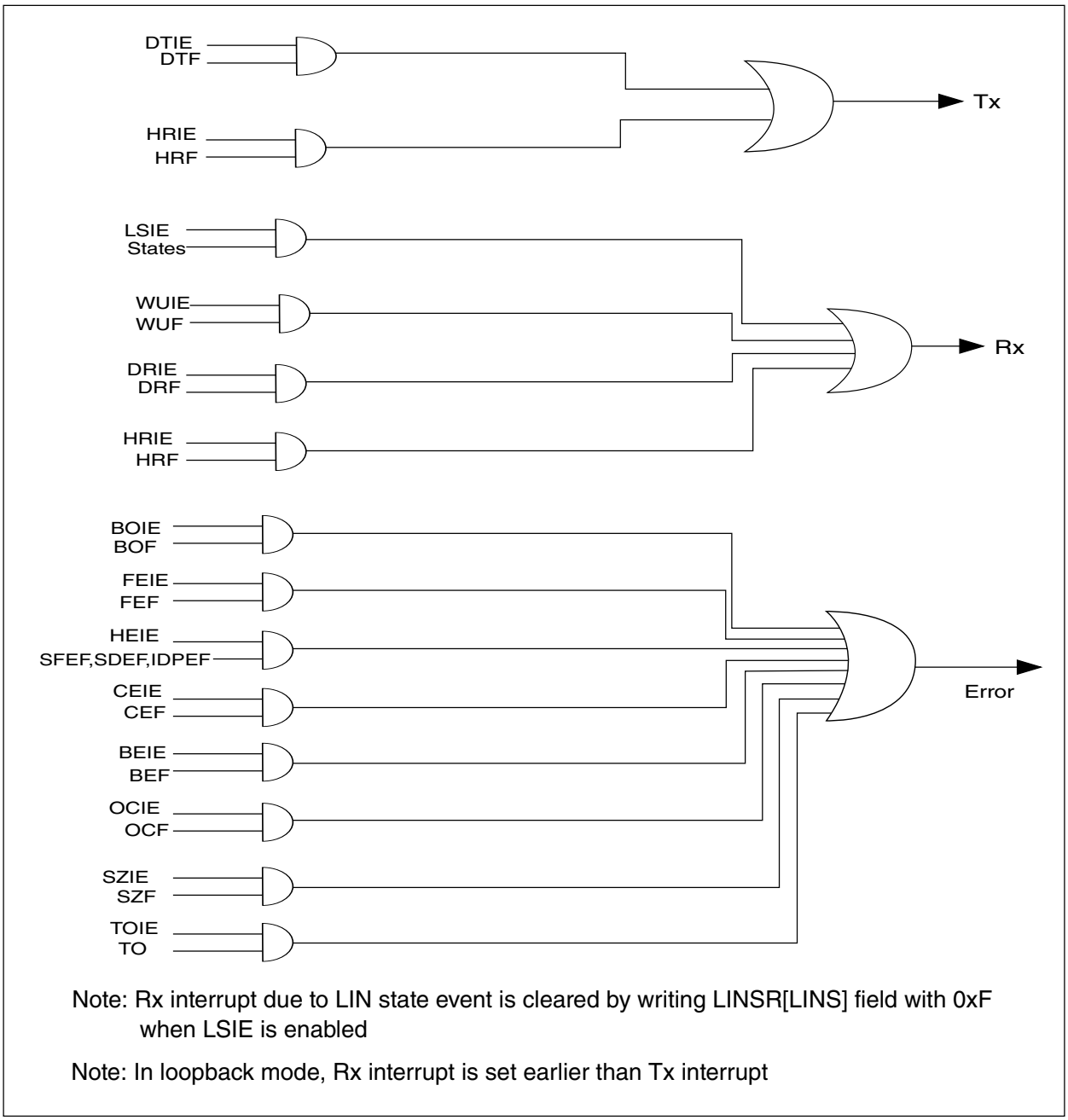
58.6.5 Interrupts

Interrupt section.

Table 58-19. Interrupts

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF orSDEF orIDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer Overrun error	BOF	BOIE	Error
UART Timeout error	TO	TOIE	Error
Lin state	Sync Del, Sync Field,Identifier field or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data Reception Complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

The following figure shows an Interrupt Flow Diagram.



Note: Rx interrupt due to LIN state event is cleared by writing LINSR[LINS] field with 0xF when LSIE is enabled

Note: In loopback mode, Rx interrupt is set earlier than Tx interrupt

Figure 58-64. Interrupt diagram

58.6.6 LINFlexD Clock Tolerance

1. **Faster receiver tolerance:**
   In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct noise and framing error free reception bit, the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

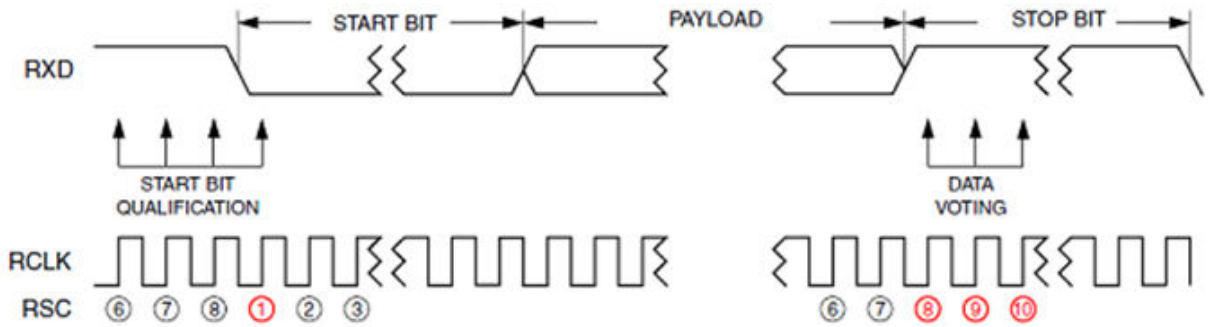


Figure 58-65. Faster receiver

2. **Slower receiver tolerance:**
   In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

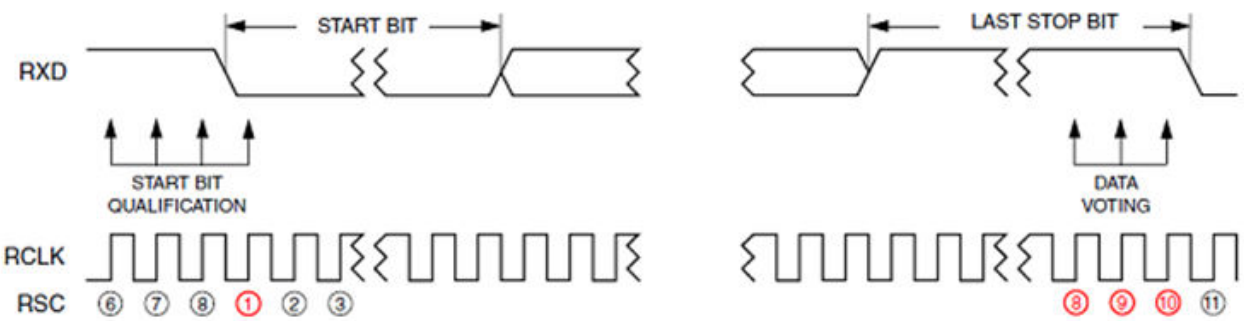


Figure 58-66. Slower receiver

Chapter 59
Inter-Integrated Circuit (I2C)

59.1 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) bus module implemented on this chip and presents the following topics:

- Introduction to I^2C
- External signal descriptions
- Memory map and register definition
- Functional description
- Initialization/application information

59.2 Introduction to I^2C

This section presents the following topics:

- Definition: I^2C module
- Advantages of the I^2C bus
- Module block diagram
- Features
- Modes of operation
- Definition: I^2C conditions

59.2.1 Definition: I2C module

The I2C module is a functional unit that provides a two-wire— serial data (SDA) and serial clock (SCL) — bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

59.2.2 Advantages of the I2C bus

The synchronous, multiple-master two-wire I2C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Does not require an external address decoder

59.2.3 Module block diagram

The following figure shows a block diagram of the I2C module.

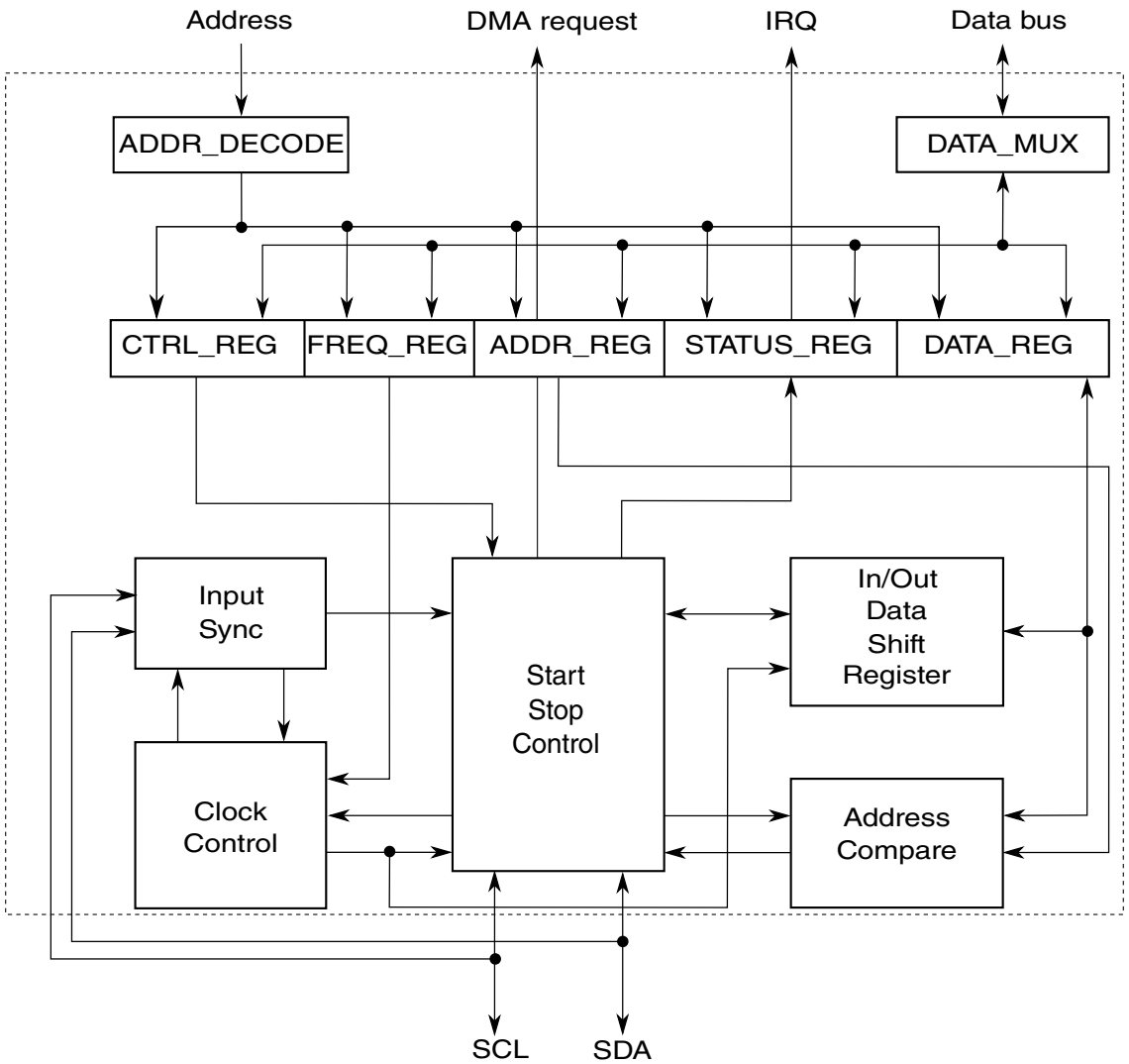


Figure 59-1. I²C block diagram

59.2.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[1]
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode

1. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

59.2.5 Modes of operation

The I^2C module supports the chip modes described in the following table.

Table 59-1. Chip modes supported by the I^2C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^2C module	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. See STOP mode.
DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging	See DEBUG mode.

In addition to chip modes, the I^2C module has several module-specific modes. These are described in the following table.

Table 59-2. Module-specific modes supported by the I²C module

Module mode	Description	Important notes
Master mode	The I²C module is the driver of the SDA line.	• Do not use the I²C module's slave address as a calling address. • The I²C module cannot be a master and a slave simultaneously.
Slave mode	The I²C module is not the driver of the SDA line.	• Enable the I²C module before a START condition from a non-I²C master is detected. • By default the I²C module performs as a slave receiver.

59.2.6 Definition: I²C conditions

The following table shows the I²C-specific conditions defined for the I²C module.

Table 59-3. I²C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in the following figure.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in the following figure.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

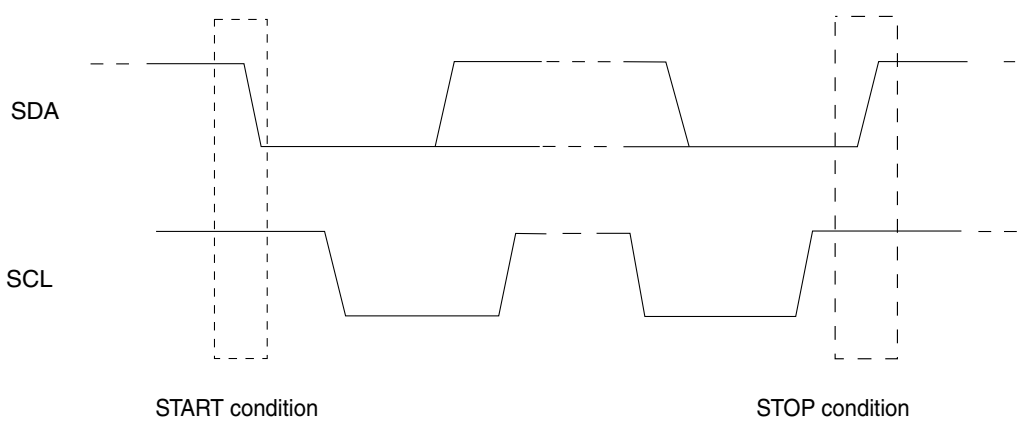


Figure 59-2. START and STOP conditions

59.3 External signal descriptions

This section presents the following topics:

- Signal overview
- Detailed external signal descriptions

59.3.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

59.3.2 Detailed external signal descriptions

The SDA and SCL signals are described in the following table.

Table 59-4. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

59.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- Register accessibility
- Register figure conventions

- I2C Bus Address Register (I2C_IBAD)
- I2C Bus Frequency Divider Register (I2C_IBFD)
- I2C Bus Control Register (I2C_IBCR)
- I2C Bus Status Register (I2C_IBSR)
- I2C Bus Data I/O Register (I2C_IBDR)
- I2C Bus Interrupt Config Register (I2C_IBIC)
- I2C Bus Debug Register (I2C_IBDBG)

59.4.1 Register accessibility

Address location 0x0007 is a reserved location, but access to this location will not generate any bus error.

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

59.4.2 Register figure conventions

The register figures show the field structure using the conventions in the following figure.

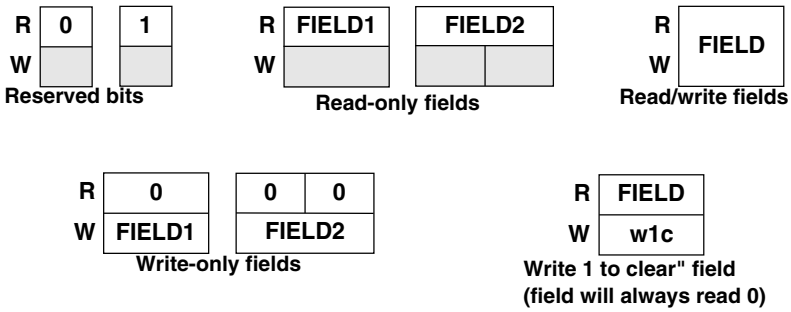


Figure 59-3. Register figure convention

The memory map for the I^{2}C module is given below. The total address for each register is the sum of the base address for the I^{2}C module and the address offset for each register.

I2C memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	I2C Bus Address Register (I2C_IBAD)	8	R/W	00h	59.4.3/2846
1	I2C Bus Frequency Divider Register (I2C_IBFD)	8	R/W	00h	59.4.4/2847
2	I2C Bus Control Register (I2C_IBCR)	8	R/W	80h	59.4.5/2847
3	I2C Bus Status Register (I2C_IBSR)	8	R/W	80h	59.4.6/2849
4	I2C Bus Data I/O Register (I2C_IBDR)	8	R/W	00h	59.4.7/2850
5	I2C Bus Interrupt Config Register (I2C_IBIC)	8	R/W	00h	59.4.8/2851
6	I2C Bus Debug Register (I2C_IBDBG)	8	R/W	00h	59.4.9/2852

59.4.3 I2C Bus Address Register (I2C_IBAD)

This register contains the address the I^{2}C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	ADR							0
Write								
Reset	0	0	0	0	0	0	0	0

I2C_IBAD field descriptions

Field	Description
0–6 ADR	Slave Address. Specific slave address to be used by the I^{2}C Bus module.

Table continues on the next page...

I2C_IBAD field descriptions (continued)

Field	Description
	NOTE: The default mode of I^2C Bus is slave mode for an address match on the bus.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.4.4 I2C Bus Frequency Divider Register (I2C_IBFD)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 1h offset = 1h

Bit	0	1	2	3	4	5	6	7
Read Write	IBC							
Reset	0	0	0	0	0	0	0	0

I2C_IBFD field descriptions

Field	Description
0–7 IBC	I-Bus Clock Rate. This field is used to prescale the bus clock for bit rate selection. See Clock rate and IBFD settings.

59.4.5 I2C Bus Control Register (I2C_IBCR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	Reserved
Write						RSTA		
Reset	1	0	0	0	0	0	0	0

I2C_IBCR field descriptions

Field	Description
0 MDIS	Module disable. This bit controls the software reset of the entire I^2C Bus module.

Table continues on the next page...

I2C_IBCR field descriptions (continued)

Field	Description
	0 The I²C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled.
1 IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I²C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I²C Bus module are enabled. An I²C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
4 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I²C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **NOTE:** Values written to this bit are only used when the I²C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5 RSTA	Repeat Start. Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. The I²C module should not attempt a repeated start when the bus is owned by another master. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable. When this bit is set, the DMA Tx and Rx lines will be asserted when the I²C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if addressed as slave conditions occur. The DMA mode is only valid when the I²C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals
7 Reserved	This field is reserved. Although this bit supports read/write access, writing to this bit is not recommended, and can produce unexpected results.

59.4.6 I2C Bus Status Register (I2C_IBSR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 3h offset = 3h

Bit	0	1	2	3	4	5	6	7
Read	TCF	IAAS	IBB	0		SRW	IBIF	RXAK
Write							w1c	
Reset	1	0	0	0	0	0	0	0

I2C_IBSR field descriptions

Field	Description
0 TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **NOTE:** This bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. **NOTE:** Software must ensure that the I^2C bus is idle by checking the IBSR[IBB] field (bus busy) before switching to master mode and attempting a START cycle. 0 Bus is Idle 1 Bus is busy
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SRW	Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master.

Table continues on the next page...

I2C_IBSR field descriptions (continued)

Field	Description
	0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
6 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: • Byte transfer complete (TCF bit set and DMAEN bit not set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

59.4.7 I2C Bus Data I/O Register (I2C_IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

NOTE

The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/ $\overline{W}$ bit (in position D0).

NOTE

When the I^2C is configured in master mode and receiving data from a slave that is transmitting data bytes on an irregular basis, the master cannot know whether the data received in the IBDR is the old latched data or the new data received from the slave. To avoid this, 2 consecutive intermittent data bytes from slave should be different.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2C_IBDR field descriptions

Field	Description
0–7 DATA	Data transmitted or received

59.4.8 I2C Bus Interrupt Config Register (I2C_IBIC)

To program BIIE = 1, you must ensure that IBCR[MDIS] = 0.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 5h offset = 5h

Bit	0	1	2	3	4	5	6	7
Read	BIIE	0	0					
Write								
Reset	0	0	0	0	0	0	0	0

I2C_IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I^2C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

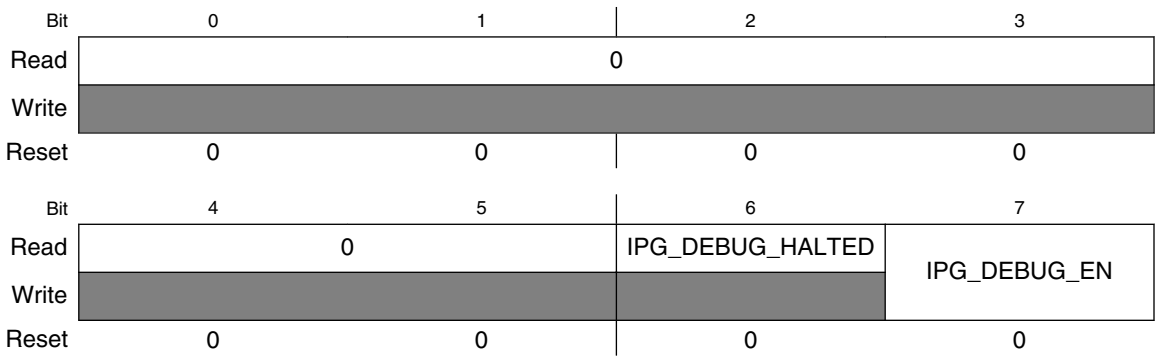
I2C_IBIC field descriptions (continued)

Field	Description
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.4.9 I2C Bus Debug Register (I2C_IBDBG)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 6h offset = 6h



I2C_IBDBG field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IPG_DEBUG_ HALTED	Debug Halted Bit: This is s status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode
7 IPG_DEBUG_EN	Debug enable bit. This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. **NOTE:** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited. 0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode

59.5 Functional description

This section presents the following topics:

- Notes about module operation
- Transactions
- Clock behavior
- Interrupts
- DEBUG mode
- DMA interface

59.5.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

59.5.2 Transactions

This section presents the following topics:

- Protocol overview
- Transaction protocol definitions
- High-level protocol steps
- START condition
- Slave address transmission
- Data transmission
- STOP condition
- Repeated START condition

59.5.2.1 Protocol overview

The following figure shows the behavior of SCL and SDA during a typical I²C transaction.

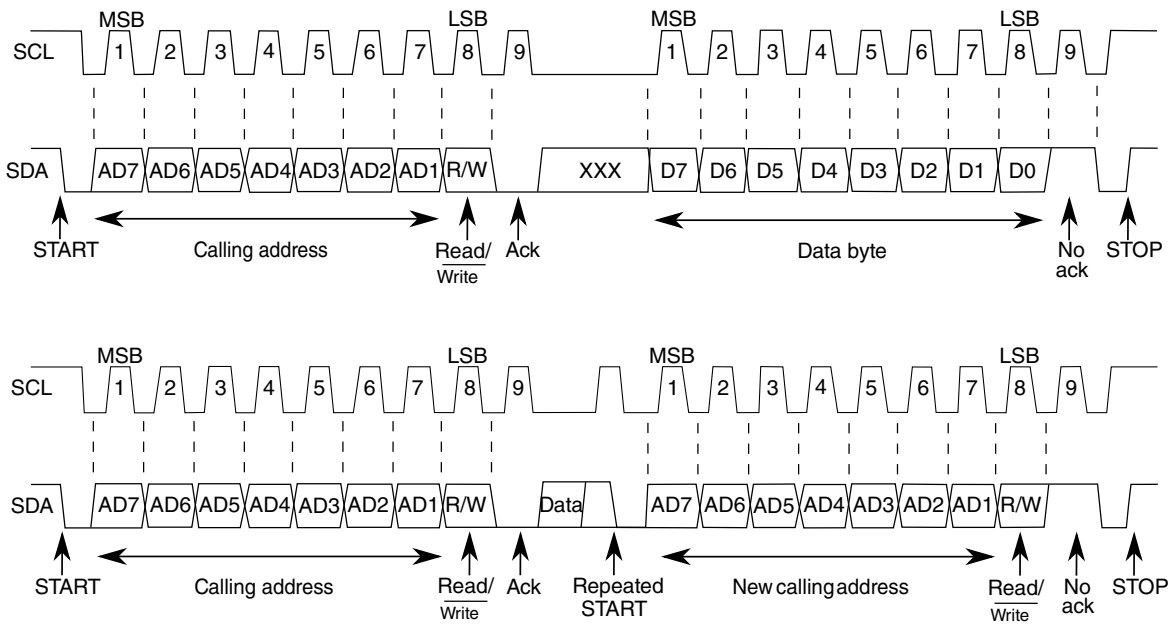


Figure 59-4. I²C transaction protocol

59.5.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 59-4.

Table 59-5. I²C definitions

Term	Definition
START	A START condition, as defined in Section 1.2.6, Definition: I²C conditions"
STOP	A STOP condition, as defined in Section 1.2.6, Definition: I²C conditions"
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in Section 1.5.2.3, I²C calling address requirements."
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: • 0=The data is being transferred from the master to the slave ("write") • 1=The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

59.5.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

59.5.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in Figure 59-4. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 59-6. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
1. START condition	1. START condition
2. Slave target or general call address transmission	2. Slave target or general call address transmission
3. Acknowledgment from slave	3. Acknowledgment from slave
4. Data transfer	4. Data transfer
5. STOP condition	5. Repeated START condition
6. (repeat Steps 1–4)	6. (repeat Steps 2–4 as needed)
	7. STOP condition.
	8. (repeat Steps 1–7)

59.5.2.5 START condition

When the bus is free, that is, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (see Definition: I²C conditions). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

59.5.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (see START condition). The process of slave address transmission is presented in the following table.

Table 59-7. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: • The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

59.5.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

59.5.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see Definition: I^2C conditions). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START condition.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

59.5.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of Figure 59-4.

59.5.3 Clock behavior

This section presents the following topics:

- Clock synchronization
- Clock stretching
- Handshaking
- Clock rate and IBFD settings

59.5.3.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached.

However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see the following figure). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

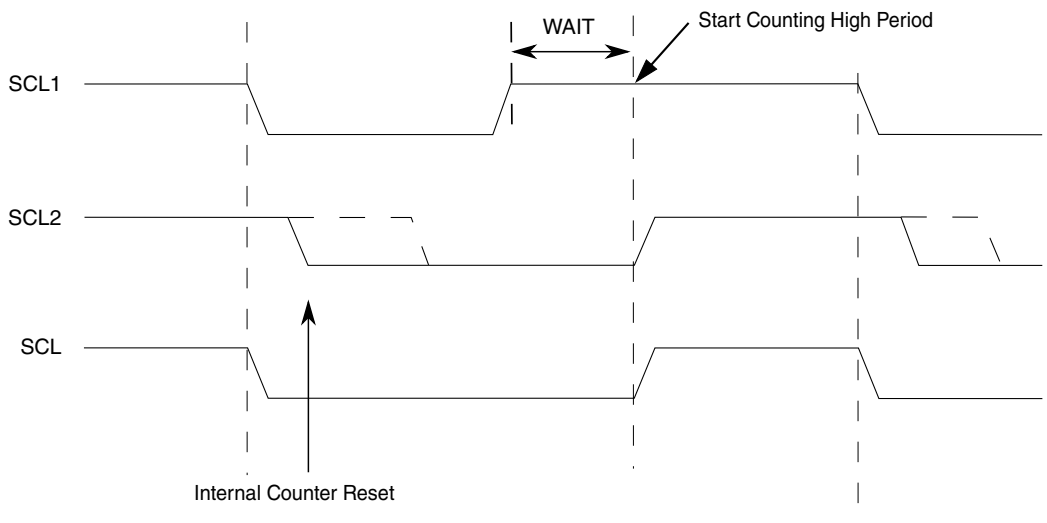


Figure 59-5. I²C bus clock synchronization

59.5.3.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

59.5.3.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

59.5.3.4 Clock rate and IBFD settings

59.5.3.4.1 Timing definitions

Table 59-8. Timing definitions relevant to clock rate and IBFD settings

Term	Definition
SCL Divider	The factor used to prescale the CPU clock for bit rate selection (see Figure 59-6 and Table 59-9)
SCL period	(CPU clock period) × (SCL Divider)
SCL Hold	The required number of CPU clocks to generate a START or STOP condition (see Figure 59-6 and Table 59-9)
SDA Hold	See Figure 59-7 and Table 59-9

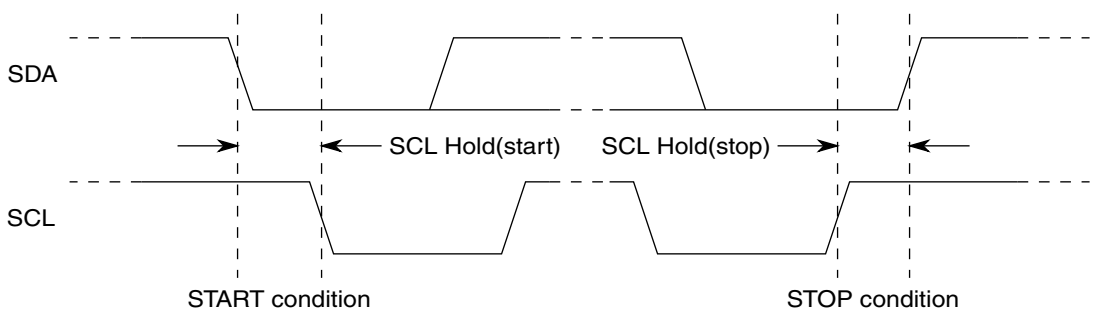


Figure 59-6. SCL Divider and SDA Hold

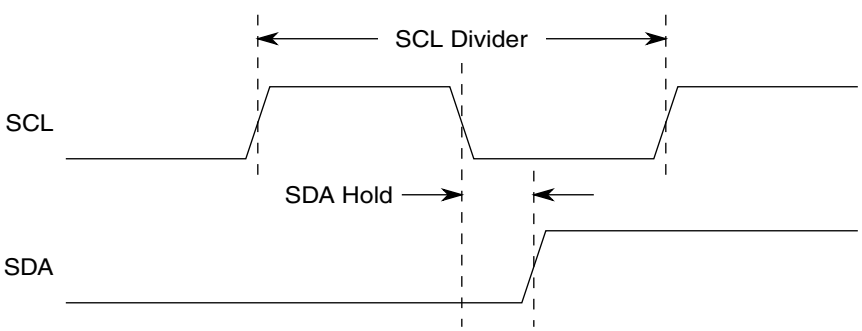


Figure 59-7. SDA Hold time

59.5.3.4.2 Divider and hold values

Table 59-9. I²C divider and hold values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
MUL=1	24	288	49	142	145

Table continues on the next page...

Table 59-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921
MUL=2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
MUL=2	4A	72	18	28	38

Table continues on the next page...

Table 59-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	34	156	162
	61	384	34	188	194
	62	448	66	220	226
	63	512	66	252	258
	64	576	98	284	290
	65	640	98	316	322
	66	768	130	380	386
	67	960	130	476	482
	68	640	66	316	322
	69	768	66	380	386
	6A	896	130	444	450
	6B	1024	130	508	514
MUL=2	6C	1152	194	572	578
	6D	1280	194	636	642
	6E	1536	258	764	770
	6F	1920	258	956	962
	70	1280	130	636	642

Table continues on the next page...

Table 59-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	71	1536	130	764	770
	72	1792	258	892	898
	73	2048	258	1020	1026
	74	2304	386	1148	1154
	75	2560	386	1276	1282
	76	3072	514	1532	1538
	77	3840	514	1916	1922
	78	2560	258	1276	1282
	79	3072	258	1532	1538
	7A	3584	514	1788	1794
	7B	4096	514	2044	2050
	7C	4608	770	2300	2306
	7D	5120	770	2556	2562
	7E	6144	1026	3068	3074
	7F	7680	1026	3836	3842
MUL=4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
MUL=4	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212

Table continues on the next page...

Table 59-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
MUL=4	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612

Table continues on the next page...

Table 59-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

59.5.4 Interrupts

This section presents the following topics:

- Interrupt vector
- Interrupt description

59.5.4.1 Interrupt vector

The I²C module uses only one interrupt vector.

Table 59-10. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I²C Interrupt	—	—	—	TCF, IAAS, IBB bits in IBSR register	When TCF or IAAS bits is set, an interrupt may be caused based on Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

59.5.4.2 Interrupt description

There are five types of internal interrupts in the I²C. The interrupt service routine can determine the interrupt type by reading the Status register.

I²C Interrupt can be generated on the following events:

- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)

- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I2C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

59.5.5 STOP mode

This mode allows the software to put the I2C module in power-down state. Once the STOP request is asserted, the I2C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I2C module enters STOP mode:

- The I2C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I2C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. See the figure below for more details.

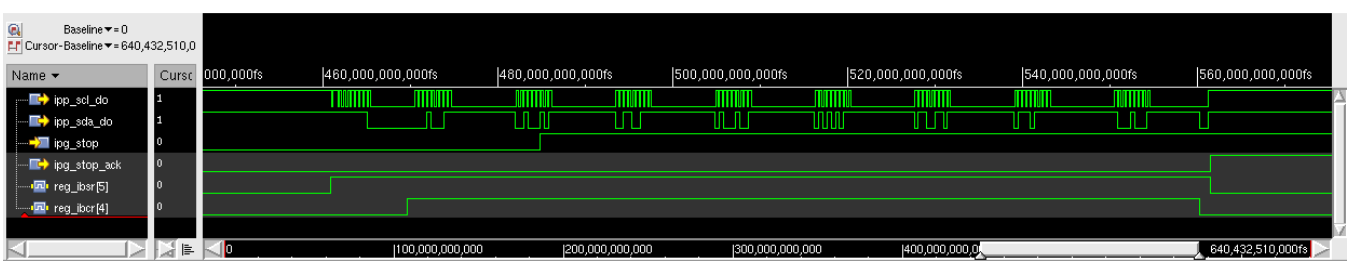


Figure 59-8. I2C stop mode behavior when master is receiving and slave is transmitting

59.5.6 DEBUG mode

This mode allows CPU to debug the I2C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I2C comes to a graceful halt after completing all the ongoing transactions. A Debug halted

signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I^{2}C module enters DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

The I^{2}C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and the debug mode was initiated by user at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. See the simulation result in Figure 59-9 for more details.

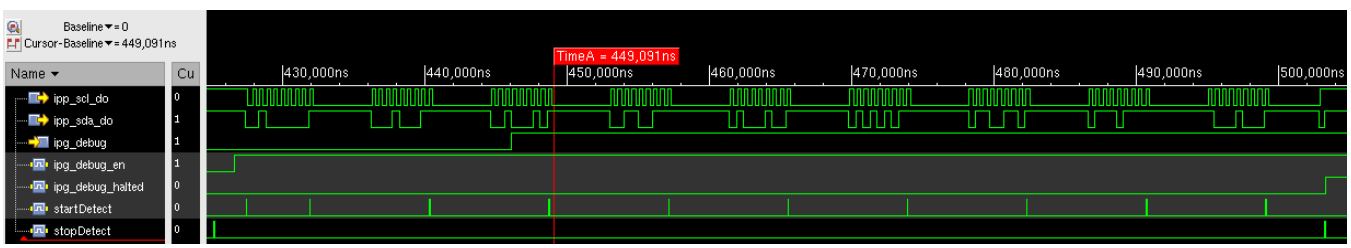


Figure 59-9. Simulation result of IPG Debug Halted in case of frame transmission

If any DMA transaction (transmit or receive) is in progress, and if the debug mode is requested while a byte is being transmitted or received, the I^{2}C module enters DEBUG mode after completing the transmission or reception of the current byte. As soon as the module exits DEBUG mode, transmission or reception of the remaining bytes resumes. Please see the simulation snapshot shown in Figure 59-10 for details where the I^{2}C is transmitting 8 bytes using DMA and DEBUG mode is requested while a second byte is being transmitted. DEBUG mode is entered after successfully transmitting the second byte (IPG Debug Halted signal asserted). As soon as the module exits DEBUG mode (IPG Debug Halted signal de-asserted), transmission resumes successfully.

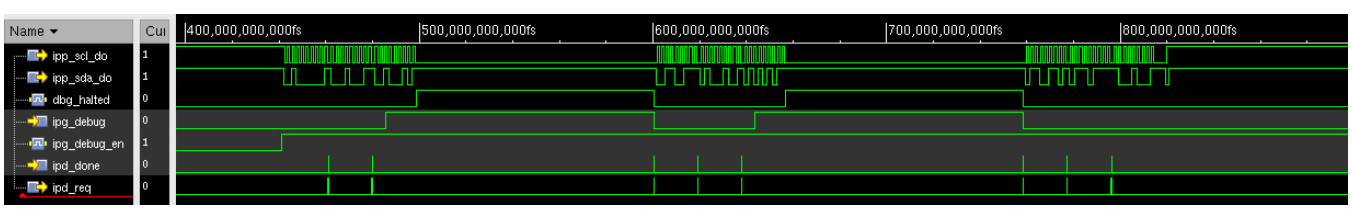


Figure 59-10. Simulation result of IPG Debug Halted in case of frame transmission using DMA

No more transaction can take place until the debug signal is de-asserted after which the I^2C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I^2C module has entered the DEBUG mode. See the following figure for more details.

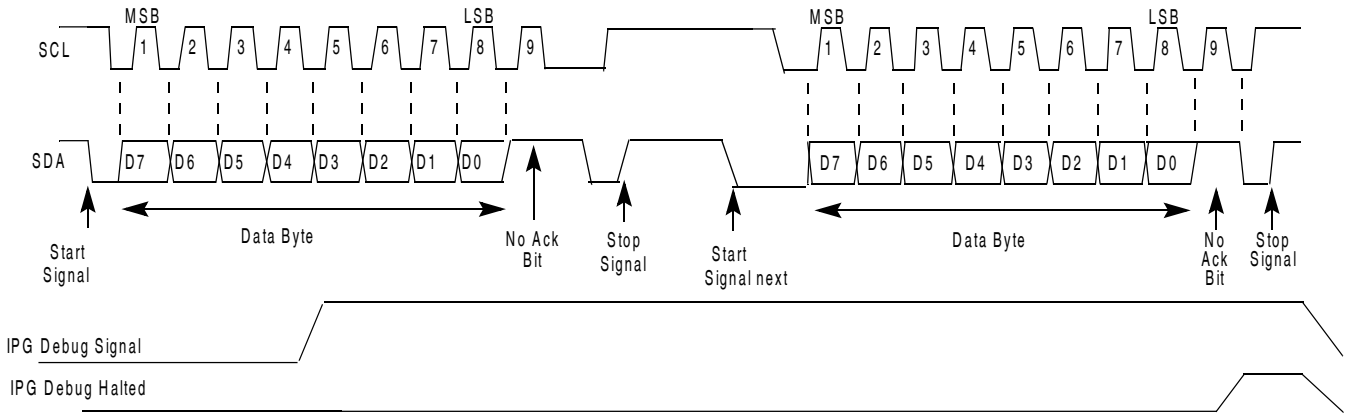


Figure 59-11. DEBUG mode

The following figure shows a case of DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters DEBUG mode.

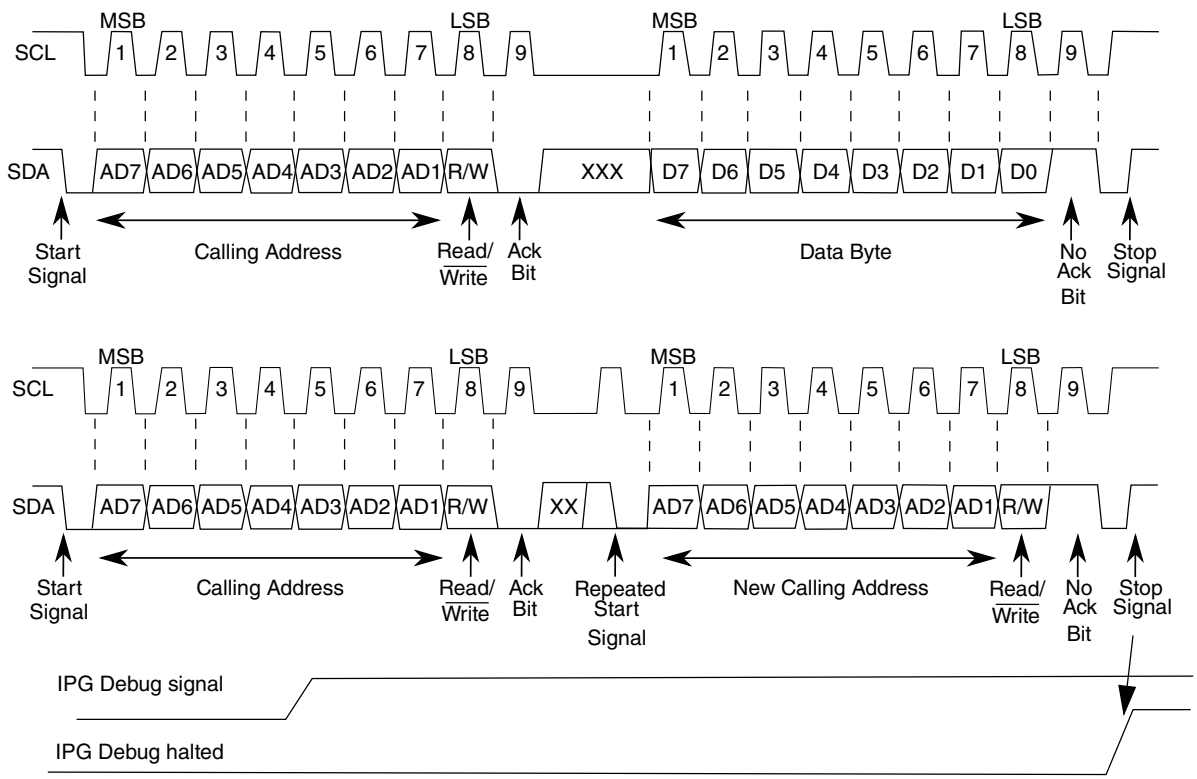


Figure 59-12. DEBUG mode with repeated start

59.5.7 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (see DMA application information). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the Initialization/application information, of this document.

59.6 Initialization/application information

This section presents the following topics:

- Recommended interrupt service flow
- General programming guidelines (for both master and slave mode)
- Programming guidelines specific to master mode
- Programming guidelines specific to slave mode
- DMA application information

59.6.1 Recommended interrupt service flow

The following figure shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

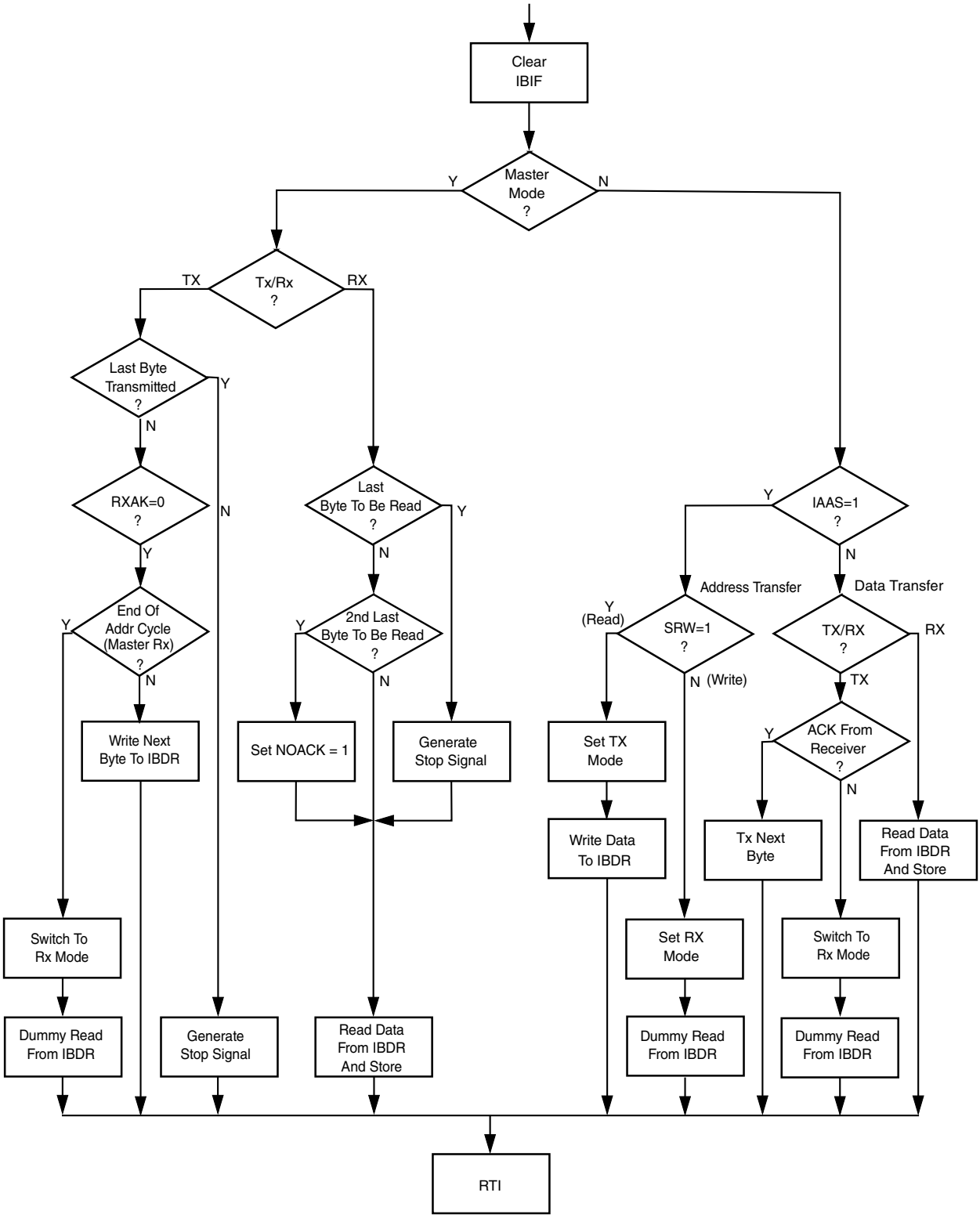


Figure 59-13. Recommended I²C interrupt service routine flowchart

59.6.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I^{2}C module in both master and slave mode. It presents the following topics:

- Initializing the I^{2}C module
- Software response after a transfer

59.6.2.1 Initializing the I^{2}C module

The following table describes how to initialize the I^{2}C module.

Table 59-11. I^{2}C initialization procedure

Step	Action
1	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from the system clock.
2	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
3	Clear the MDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^{2}C interface system.
4	Use I2C Bus Control Register (I2C_IBCR) to select Master/Slave mode, Transmit/Receive mode, and whether interrupts are enabled or disabled.
5	(Optional) Use I2C Bus Interrupt Config Register (I2C_IBIC) to further refine the interrupt behavior.

59.6.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^{2}C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I^{2}C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^{2}C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

59.6.3 Programming guidelines specific to master mode

This section presents the following topics:

- Generating START
- Transmit/receive sequence
- Generating STOP
- Generating repeated START

59.6.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1    // master and transmit mode, that is,
                                    // generate start condition
IBDR = calling_address              // send the calling address to the data register
while (bit 5, IBSR ==0)             // wait in loop for IBB flag to be set
```

59.6.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 59-12. Master transmit sequence

Step	Action
a	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Use I2C Bus Control Register (I2C_IBCR) to select Master mode, Transmit mode, and interrupt enable.
d	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
e	Write data to I2C Bus Data I/O Register (I2C_IBDR).
f	Observe changes in the TCF field of I2C Bus Status Register (I2C_IBSR) : • When IBSR[TCF] becomes 0, the transfer is in progress. • When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the transfer completed. • If RXAK = 1, a No Acknowledge condition occurred. • If IBB = 0, the bus transitioned from Busy to Idle state. **NOTE:** You can ignore Address Detect (IAAS = 1) for master mode (it is valid only for slave mode).
i	Examine the RXAK field in I2C Bus Status Register (I2C_IBSR) for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 59-13. Master receive sequence

Step	Action
a	Follow steps a through i in Table 59-12 for address dispatch.
b	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
c	Write 0 to the TXRX field in I2C Bus Control Register (I2C_IBCR) to select Receive mode.

Table continues on the next page...

Table 59-13. Master receive sequence (continued)

Step	Action
d	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
e	Wait until the TCF field in I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
f	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
g	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the slave.

Table 59-14. Slave transmit sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
d	Write data to I2C Bus Data I/O Register (I2C_IBDR).
e	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
f	Wait until the RXAK field in I2C Bus Status Register (I2C_IBSR) becomes 0.
g	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
h	Repeat steps d through g for the next consecutive data transfers.

Table 59-15. Slave receive sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field of I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
d	Write 0 to the IBIF field of I2C Bus Status Register (I2C_IBSR).
e	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
f	Wait until the TCF field of I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
g	Wait until the IBIF field of I2C Bus Status Register (I2C_IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
i	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the master.

59.6.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or     // check to see if all data bytes have been transmitted
  (bit 0, IBSR == 1) {    // or if no ACK generated
    clear bit 5, IBCR     // generate stop condition
  }
else {
 IBDR = data_to_transmit  // write byte of data to DATA register
   tx_count --            // decrement counter
 }                        // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --             // decrease the rx counter
if (rx_count ==1)       // 2nd last byte to be read ?
 bit 3, IBCR = 1        // disable ACK
 if (rx_count == 0)     // last byte to be read ?
 bit 5, IBCR = 0        // generate stop signal
else
data_received = IBDR    // read RX data and store
```

59.6.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1             // generate another start (restart)
IBDR == calling_address     // transmit the calling address
```

59.6.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to

the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an "end of data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

59.6.5 DMA application information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMAEN field in I2C Bus Control Register (I2C_IBCR) is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer. The DMAEN field in I2C Bus Control Register (I2C_IBCR) works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, depending on the setting of the DMAEN field. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place.

The sections present the following topics:

- DMA mode, master transmit
- DMA mode, master reception
- Exiting DMA mode, system requirement considerations

59.6.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit "n" data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

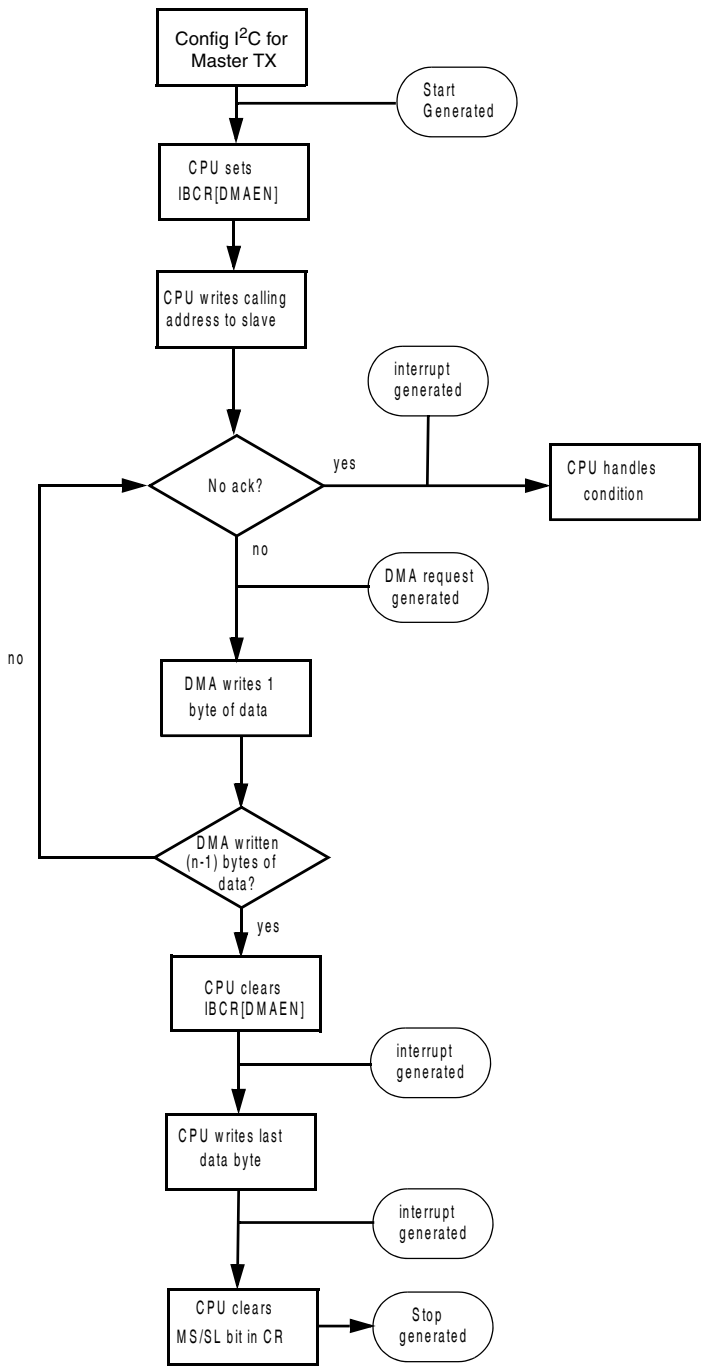


Figure 59-14. Flow-Chart of DMA mode master transmit

59.6.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive "n" data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

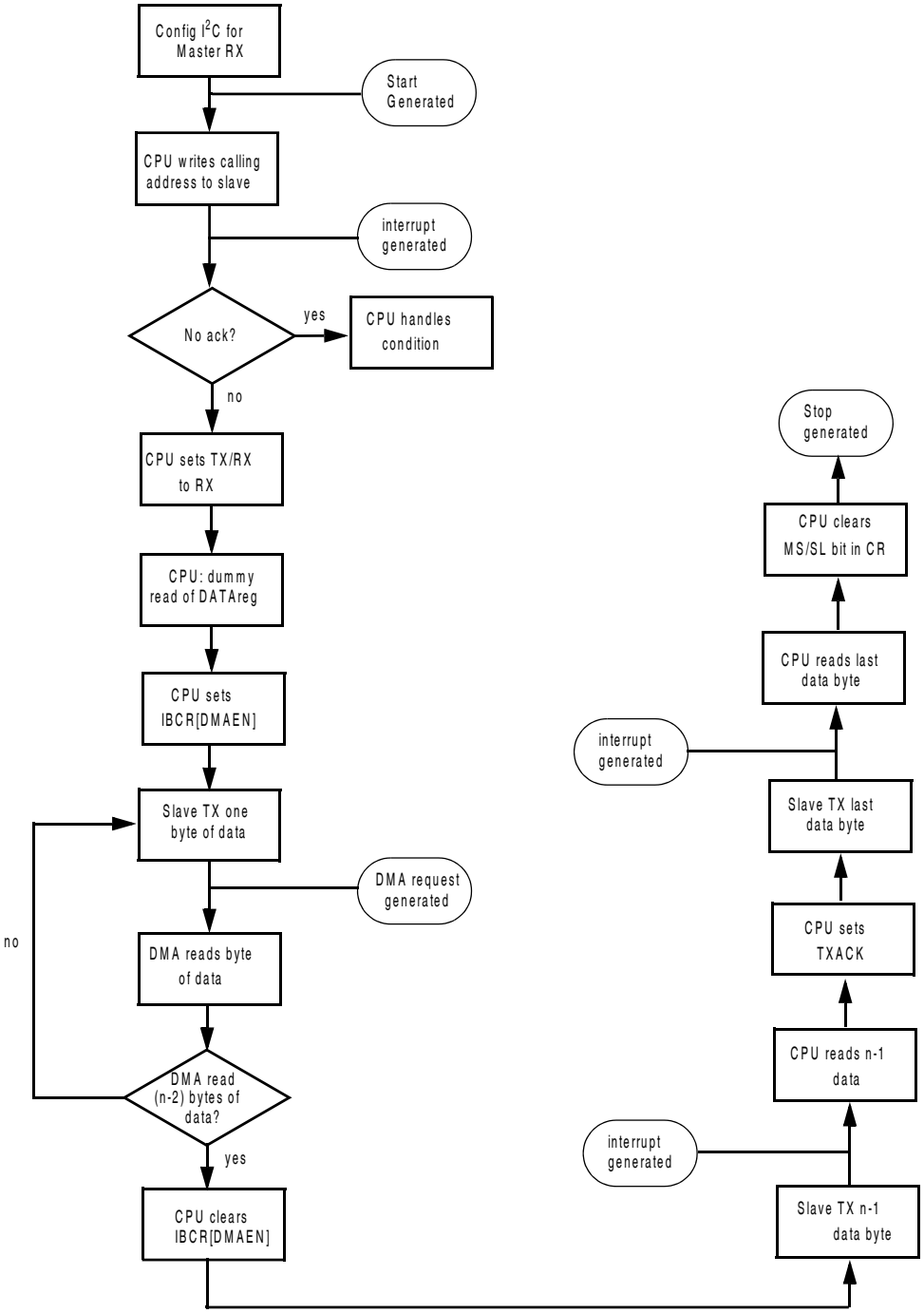


Figure 59-15. Flow-Chart of DMA mode master receive

59.6.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I^2C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For 'fast reaction' the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

$$T_R = 8 \times T_{SCL}$$

In fast mode, with 400kbit/s, T_{SCL} is 2.5µs, so T_R is 20µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act

accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note

The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

   Note

   Here you have to make sure on system level that no higher priority DMA requests occur between the two linked TCDs as those could again create a scenario of slow reaction.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload the initial TCD upon completion, bringing the system back into a "ready-for-I^2C-transfer" state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note

Here you have to make sure at system level that no higher priority DMA requests occur during the scatter/gather process, as those could again create a scenario of slow reaction.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/16 MHz = 250 ns

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

  $8 \times T_{AHD}$ = 8/32 MHz = 250 ns

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/150 MHz = 26.6 ns

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

  $4 \times T_{AHD}$ = 4/150 MHz = 26.6 ns

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

Chapter 60
FlexRay Communication Controller (FlexRay)

60.1 Chip-specific FlexRay information

60.1.1 FlexRay Configuration

The FlexRay communications system is designed to provide high-speed deterministic distributed control for advanced automotive applications. Its dual-channel architecture supporting data rates up to 10 Mbit/s per channel offers system-wide redundancy that supports the reliability requirements of enhanced availability and safety system.

60.1.1.1 FlexRay signals pin assignment

This table lists the pin names for the FlexRay signals that are connected to pins on the boundary of the chip.

Table 60-1. Flexray pin names

Signal description	Pin name
FR_A_RX (Receive Data Channel A)	CA_RX
FR_A_TX (Transmit Data Channel A)	TXD_A
$\overline{\text{FR\_A\_TX\_EN}}$ (Transmit Enable Channel A)	$\overline{\text{TXEN\_A}}$
FR_B_RX (Receive Data Channel B)	CB_RX
FR_B_TX (Transmit Data Channel B)	CB_TX
$\overline{\text{FR\_B\_TX\_EN}}$ (Transmit Enable Channel B)	$\overline{\text{TXEN\_B}}$
FR_DBG[0] (Debug Strobe Signal 0)	DEBUG_0
FR_DBG[1] (Debug Strobe Signal 1)	DEBUG_1
FR_DBG[2] (Debug Strobe Signal 2)	DEBUG_2
FR_DBG[3] (Debug Strobe Signal 3)	DEBUG_3

60.2 Introduction

This section provides a high-level summary of module features.

60.2.1 Reference

The following documents are referenced.

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*[1]
  - Refer to this document for the all of the FlexRay related informaton including the configuration parameters and the allowed parameter ranges.
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0*

60.2.2 Glossary

This section provides a list of terms used in this chapter.

Table 60-2. List of terms

Term	Definition
BCU	Buffer Control Unit. Handles message buffer access.
BMIF	Bus Master Interface. Provides master access to FlexRay memory area.
CC	Communication Controller
CDC	Clock Domain Crosser
CHI	Controller Host Interface
Cycle length in µT	The actual length of a cycle in µT for the ideal controller (+/- 0 ppm)
EBI	External Bus Interface
FlexRay Memory Area	Memory area to store the physical message buffer payload data, frame header, frame and slot status, and synchronization frame related tables.
FSS	Frame Start Sequence
HIF	Host Interface. Provides host access to controller.
Host	The FlexRay CC host MCU
Keyslot	Key slot is used to transmit the startup frame, sync frame, or designated single slot frame.

Table continues on the next page...

1. The FlexRay Specifications have been developed for automotive applications.The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Table 60-2. List of terms (continued)

Term	Definition
LUT	Look Up Table. Stores message buffer header index value.
LRAM	Look Up Table RAM. Module internal memory to store message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers.
MB	Message Buffer
Mini-slot	An interval of time within the dynamic segment of the communication cycle that is of constant duration (in terms of macroticks) and that is used by the synchronized FTDMA media access scheme to manage media arbitration
MBIDX	Message Buffer Index: the position of a header field entry within the header area. If the header area is accessed as an array, this is the same as the array index of the entry.
MBNum	Message Buffer Number: Position of message buffer configuration registers within the register map. For example, Message Buffer Number 5 corresponds to the MBCCS5 register.
MCU	Microcontroller Unit
µT	Microtick
MT	Macrotick
MTS	Media Access Test Symbol
message frame	Frame with *Null Frame Indicator* set to 1
normal frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 0
null frame	frame with *Null Frame Indicator* set to 0
NIT	Network Idle Time
PE	Protocol Engine
POC	Protocol Operation Control. Each state of the POC is denoted by POC:state
Rx	Reception
Slot	An interval of time during which access to a communication channel is granted exclusively to a specific node for the transmission of a frame with a frame ID corresponding to the slot.
SEQ	Sequencer Engine
SU	Status update
SECDED	Single-bit error correction, double-bit error detection
System Memory	Memory that contains the FlexRay Memory Area.
System Bus	Bus that connects the controller and System Memory
sync frame	null frame or message frame with *Sync Frame Indicator* set to 1
startup frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 1
TCU	Time Control Unit
Tx	Transmission

60.2.3 Overview

The CC is a FlexRay communication controller that implements the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

The CC has three main components:

- *Controller host interface (CHI)*
- *Protocol engine (PE)*
- *Clock domain crossing unit (CDC)*

A block diagram of the CC with its surrounding modules is given in the below figure.

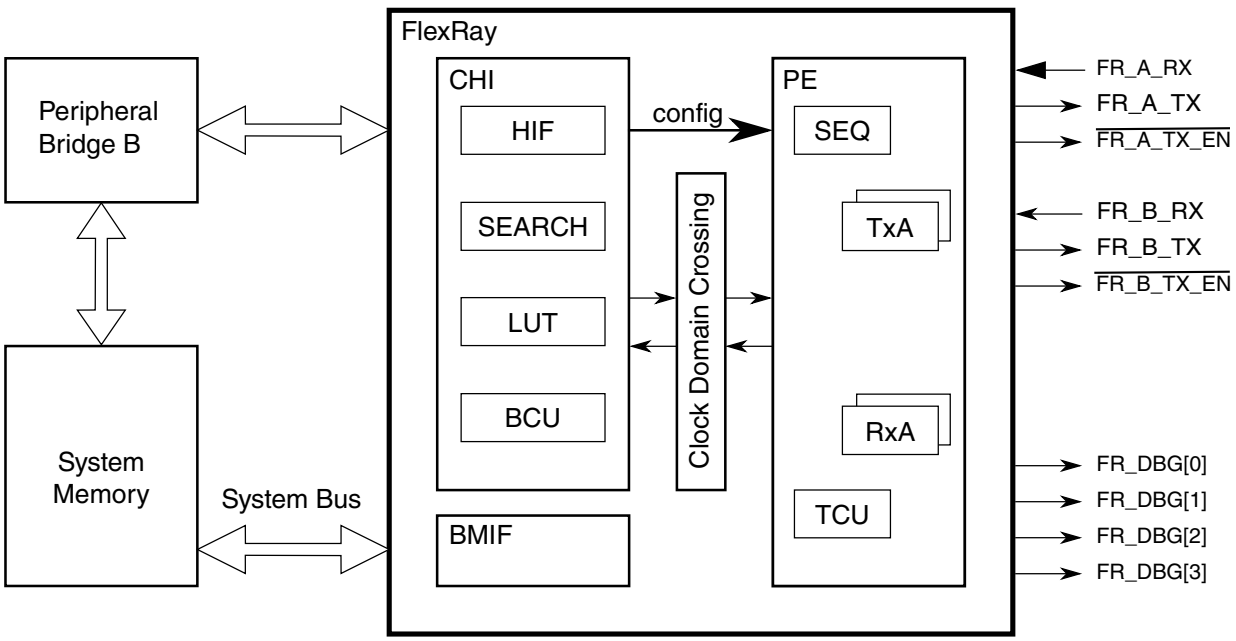


Figure 60-1. FlexRay block diagram

The protocol engine has two transmitter units TxA and TxB and two receiver units RxA and RxB for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the PE is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, and the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the CHI clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the System Memory Base Address Register (FR_SYMBADHR and FR_SYMBADLR). All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note

The CC does not provide a memory protection scheme for the FlexRay memory area.

60.2.4 Features

The CC provides the following features:

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A compliant protocol implementation*
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0 compliant bus driver interface*
- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- Dual channel support
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 128 configurable message buffers with
  - individual frame ID filtering

    - individual channel ID filtering
    - individual cycle counter filtering
- Message buffer header, status and payload data stored in dedicated FlexRay memory area
    - allows for flexible and efficient message buffer implementation
    - consistent data access ensured by means of buffer locking scheme
    - application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0 up to 254 bytes
- Two independent message buffer segments with configurable size of payload data section
    - each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
    - applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
    - receive message buffer
    - transmit message buffer
- Individual message buffer reconfiguration supported
    - means provided to safely disable individual message buffers
    - disabled message buffers can be reconfigured
- Two independent receive FIFOs
    - one receive FIFO per channel
    - up to 255 entries for each FIFO
    - global frame ID filtering, based on both value/mask filters and range filters
    - global channel ID filtering
    - global message ID filtering for the dynamic segment

- 4 configurable slot error counters
- 4 dedicated slot status indicators
  - used to observe slots without using receive message buffers
- Measured value indicators for the clock synchronization
  - internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- 1 absolute timer
- 1 timer that can be configured to absolute or relative
- Error correction and error detection (SECDED ECC) for protocol engine data RAM
- Error detection (SECDED ECC) for CHI lookup table RAM

60.2.5 Modes of Operation

This section describes the basic operational power modes of the CC.

60.2.5.1 Disabled Mode

The CC enters the Disabled Mode during hard reset or when the host clears the ‘MEN’ field in the Module Configuration Register (FR_MCR). The host can clear this field by writing ‘0’ and only when the module is in POC:default config mode. The Disabled mode can be checked by reading the module enable field, MEN, in the Module Configuration Register (FR_MCR).

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled Mode* can be accessed for writing as stated in the Memory Map and Register Description section.

The application configures the CC by accessing the configuration bits and fields in the Module Configuration Register (FR_MCR) as described in Module Initialization .

60.2.5.1.1 Leave Disabled Mode

The CC leaves the Disabled Mode and enters the Normal Mode, when the application writes 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

Note

Once the CC is enabled, it can only be disabled during POC: default config.

60.2.5.2 Normal Mode

In this mode the CC is fully functional. The CC indicates that it is in Normal Mode by asserting the module enable bit MEN in the Module Configuration Register (FR_MCR).

60.2.5.2.1 Enter Normal Mode

This mode is entered when the application requests the CC to leave the Disabled Mode. If the Normal Mode was entered by leaving the Disabled Mode, the application has to perform the protocol initialization described in Protocol Initialization to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB bits in the Module Configuration Register (FR_MCR), the corresponding FlexRay bus driver ports are enabled and driven.

60.3 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in the following table and described in detail in the Detailed Signal Descriptions section below.

NOTE

Please refer the device configuration of the reference manual for the exact module pin names.

NOTE

The off chip signals FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 60-3. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\text{FR\_A\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\text{FR\_B\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

60.3.1 Detailed Signal Descriptions

This section provides a detailed description of the CC signals, connected to external pins.

60.3.1.1 FR_A_RX — Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

60.3.1.2 FR_A_TX — Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

60.3.1.3 $\overline{\text{FR\_A\_TX\_EN}}$ — Transmit Enable Channel A

The $\overline{\text{FR\_A\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

60.3.1.4 FR_B_RX — Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

60.3.1.5 FR_B_TX — Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

60.3.1.6 $\overline{\text{FR\_B\_TX\_EN}}$ — Transmit Enable Channel B

The $\overline{\text{FR\_B\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

60.3.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] — Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection refer to the Strobe Signal Support.

60.4 Controller Host Interface Clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

The first constraint corresponds to the number of utilized message buffers and is specified in Number of Usable Message Buffers.

The second constraint corresponds to the value of the TIMEOUT field in the System Memory Access Time-Out Register (FR_SYMATOR) and is specified in Configure System Memory Access Time-Out Register (FR_SYMATOR).

60.5 Protocol Engine Clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select bit CLKSEL in the Module Configuration Register (FR_MCR).

Note

See the Device clocking details for the exact clock sources used.

60.5.1 Oscillator Clocking

If the protocol engine is clocked by the internal crystal oscillator, a crystal or CMOS compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version* 2.1 Rev A.

60.5.2 PLL Clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately and provided to the PE.

For more details, see the clocking chapter of the device reference manual.

60.6 Register Descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities.

Table 60-4. Register Access Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
R*	Reserved bit or field, will not be changed. Application must not write any value different from the reset value.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 60-5. Register Field Types

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.

Table continues on the next page...

Table 60-5. Register Field Types (continued)

Convention	Description
Reset Value	
0	Resets to zero.
1	Resets to one.
-	Not defined after reset and not affected by reset.

60.6.1 Register Reset

All registers except the Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*), Message Buffer Frame ID Register (FR_MBFIDR*n*), and Message Buffer Index Register (FR_MBIDXR*n*) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are mentioned in the detailed description of the register. The additional reset conditions are explained in the table below.

Table 60-6. Additional Register Reset Conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes to RUN command "0101" to the POCCMD field in the Protocol Operation Control Register (FR_POCR).
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants the disable to the application by clearing the FR_MBCCSRn[EDS] bit.

60.6.2 Register Write Access

This section describes the write access restriction terms that apply to all registers.

60.6.2.1 Register Write Access Restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the table below. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any

notification. The values of the bits or fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 60-7. Register Write Access Restrictions

Condition	Indication	Description
Any Time	-	No write access restriction.
Disabled Mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled Mode.
Normal Mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal Mode.
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when Protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSR[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.

60.6.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

60.6.2.3 Internal Register Access

The following memory mapped registers are used to access multiple internal registers.

- *Strobe Signal Control Register (FR_STBSCR)*
- *Slot Status Selection Register (FR_SSSR)*
- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Configuration Data*

Each of these memory mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If the WMD bit is set to 0 during the write access, all fields of the internal register are updated.

If the WMD bit set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

60.7 Memory map and register definition

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU. Address offset - 0x0Ah,0xDEh-0xE0hand 0xE2h-0xE4hshould not be accessed by application as corresponding feature/s are not available. Therefore, transfer error will not be generated at these offset/s.

NOTE

The following registers are 16-bit write accessible only.

Strobe Signal Control Register (FR_STBSCR)

PE DRAM Access Register (FR_PEDRAR)

PE DRAM Data Register (FR_PEDRDR)

Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Slot Status Selection Register (FR_SSSR)

Slot Status Counter Condition Register (FR_SSCCR)

Receive Shadow Buffer Index Register (FR_RSBIR)

Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)

Message Buffer Frame ID Register (FR_MBFIDRn)

Message Buffer Index Register (FR_MBIDXRn)

Message Buffer Data Field Offset Register (FR_MBDORn)

LRAM ECC Error Test Register (FR_LEETRn)

FR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Version Register (FR_MVR)	16	R	A568h	60.7.1/2932

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2	Module Configuration Register (FR_MCR)	16	R/W	0000h	60.7.2/2932
4	System Memory Base Address High Register (FR_SYMBADHR)	16	R/W	0000h	60.7.3/2935
6	System Memory Base Address Low Register (FR_SYMBADLR)	16	R/W	0000h	60.7.4/2935
8	Strobe Signal Control Register (FR_STBSCR)	16	R/W	0000h	60.7.5/2936
C	Message Buffer Data Size Register (FR_MBDSR)	16	R/W	0000h	60.7.6/2938
E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	16	R/W	7F7Fh	60.7.7/2939
10	PE DRAM Access Register (FR_PEDRAR)	16	R/W	0000h	60.7.8/2940
12	PE DRAM Data Register (FR_PEDRDR)	16	R/W	0000h	60.7.9/2941
14	Protocol Operation Control Register (FR_POCR)	16	R/W	0000h	60.7.10/ 2941
16	Global Interrupt Flag and Enable Register (FR_GIFER)	16	R/W	0000h	60.7.11/ 2943
18	Protocol Interrupt Flag Register 0 (FR_PIFR0)	16	R/W	0000h	60.7.12/ 2946
1A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	16	R/W	0000h	60.7.13/ 2948
1C	Protocol Interrupt Enable Register 0 (FR_PIER0)	16	R/W	0000h	60.7.14/ 2950
1E	Protocol Interrupt Enable Register 1 (FR_PIER1)	16	R/W	0000h	60.7.15/ 2952
20	CHI Error Flag Register (FR_CHIERFR)	16	R/W	0000h	60.7.16/ 2953
22	Message Buffer Interrupt Vector Register (FR_MBIVEC)	16	R	0000h	60.7.17/ 2956
24	Channel A Status Error Counter Register (FR_CASERCR)	16	R	0000h	60.7.18/ 2957
26	Channel B Status Error Counter Register (FR_CBSERCR)	16	R	0000h	60.7.19/ 2957
28	Protocol Status Register 0 (FR_PSR0)	16	R	0000h	60.7.20/ 2958
2A	Protocol Status Register 1 (FR_PSR1)	16	R	0000h	60.7.21/ 2960
2C	Protocol Status Register 2 (FR_PSR2)	16	R	0000h	60.7.22/ 2961
2E	Protocol Status Register 3 (FR_PSR3)	16	R/W	0000h	60.7.23/ 2963
30	Macrotick Counter Register (FR_MTCTR)	16	R	0000h	60.7.24/ 2965
32	Cycle Counter Register (FR_CYCTR)	16	R	0000h	60.7.25/ 2966

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Slot Counter Channel A Register (FR_SLTCTAR)	16	R	0000h	60.7.26/ 2966
36	Slot Counter Channel B Register (FR_SLTCTBR)	16	R	0000h	60.7.27/ 2967
38	Rate Correction Value Register (FR_RTCORVR)	16	R	0000h	60.7.28/ 2967
3A	Offset Correction Value Register (FR_OFCORVR)	16	R	0000h	60.7.29/ 2968
3C	Combined Interrupt Flag Register (FR_CIFR)	16	R	0000h	60.7.30/ 2969
3E	System Memory Access Time-Out Register (FR_SYMATOR)	16	R/W	0006h	60.7.31/ 2970
40	Sync Frame Counter Register (FR_SFCNTR)	16	R	0000h	60.7.32/ 2971
42	Sync Frame Table Offset Register (FR_SFTOR)	16	R/W	0000h	60.7.33/ 2971
44	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	16	R/W	0000h	60.7.34/ 2972
46	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	16	R/W	0000h	60.7.35/ 2974
48	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	16	R/W	0000h	60.7.36/ 2974
4A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	16	R/W	0000h	60.7.37/ 2975
4C	Network Management Vector Register (FR_NMVR0)	16	R	0000h	60.7.38/ 2975
4E	Network Management Vector Register (FR_NMVR1)	16	R	0000h	60.7.38/ 2975
50	Network Management Vector Register (FR_NMVR2)	16	R	0000h	60.7.38/ 2975
52	Network Management Vector Register (FR_NMVR3)	16	R	0000h	60.7.38/ 2975
54	Network Management Vector Register (FR_NMVR4)	16	R	0000h	60.7.38/ 2975
56	Network Management Vector Register (FR_NMVR5)	16	R	0000h	60.7.38/ 2975
58	Network Management Vector Length Register (FR_NMVLR)	16	R/W	0000h	60.7.39/ 2976
5A	Timer Configuration and Control Register (FR_TICCR)	16	R/W	0000h	60.7.40/ 2976
5C	Timer 1 Cycle Set Register (FR_TI1CYSR)	16	R/W	0000h	60.7.41/ 2978
5E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	16	R/W	0000h	60.7.42/ 2979

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)	16	R/W	0000h	60.7.43/ 2979
60	Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)	16	R/W	0000h	60.7.44/ 2980
62	Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)	16	R/W	0000h	60.7.45/ 2980
62	Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)	16	R/W	0000h	60.7.46/ 2981
64	Slot Status Selection Register (FR_SSSR)	16	R/W	0000h	60.7.47/ 2982
66	Slot Status Counter Condition Register (FR_SSCCR)	16	R/W	0000h	60.7.48/ 2983
68	Slot Status Register (FR_SSR0)	16	R	0000h	60.7.49/ 2985
6A	Slot Status Register (FR_SSR1)	16	R	0000h	60.7.49/ 2985
6C	Slot Status Register (FR_SSR2)	16	R	0000h	60.7.49/ 2985
6E	Slot Status Register (FR_SSR3)	16	R	0000h	60.7.49/ 2985
70	Slot Status Register (FR_SSR4)	16	R	0000h	60.7.49/ 2985
72	Slot Status Register (FR_SSR5)	16	R	0000h	60.7.49/ 2985
74	Slot Status Register (FR_SSR6)	16	R	0000h	60.7.49/ 2985
76	Slot Status Register (FR_SSR7)	16	R	0000h	60.7.49/ 2985
78	Slot Status Counter Register (FR_SSCR0)	16	R	0000h	60.7.50/ 2987
7A	Slot Status Counter Register (FR_SSCR1)	16	R	0000h	60.7.50/ 2987
7C	Slot Status Counter Register (FR_SSCR2)	16	R	0000h	60.7.50/ 2987
7E	Slot Status Counter Register (FR_SSCR3)	16	R	0000h	60.7.50/ 2987
80	MTS A Configuration Register (FR_MTSACFR)	16	R/W	0000h	60.7.51/ 2987
82	MTS B Configuration Register (FR_MTSBCFR)	16	R/W	0000h	60.7.52/ 2988
84	Receive Shadow Buffer Index Register (FR_RSBIR)	16	R/W	0000h	60.7.53/ 2989
86	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	16	R/W	0000h	60.7.54/ 2990

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Receive FIFO Start Index Register (FR_RFSIR)	16	R/W	0000h	60.7.55/ 2991
8A	Receive FIFO Depth and Size Register (FR_RFDSR)	16	R/W	0000h	60.7.56/ 2991
8C	Receive FIFO A Read Index Register (FR_RFARIR)	16	R	0000h	60.7.57/ 2992
8E	Receive FIFO B Read Index Register (FR_RFBRIR)	16	R	0000h	60.7.58/ 2992
90	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	16	R/W	0000h	60.7.59/ 2993
92	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	16	R/W	0000h	60.7.60/ 2993
94	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	16	R/W	0000h	60.7.61/ 2994
96	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	16	R/W	0000h	60.7.62/ 2994
98	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	16	R/W	0000h	60.7.63/ 2995
9A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	16	R/W	0000h	60.7.64/ 2996
9C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	16	R	0000h	60.7.65/ 2997
9E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	16	R	0000h	60.7.66/ 2998
A0	Protocol Configuration Register 0 (FR_PCR0)	16	R/W	0000h	60.7.67/ 2998
A2	Protocol Configuration Register 1 (FR_PCR1)	16	R/W	0000h	60.7.68/ 3001
A4	Protocol Configuration Register 2 (FR_PCR2)	16	R/W	0000h	60.7.69/ 3001
A6	Protocol Configuration Register 3 (FR_PCR3)	16	R/W	0000h	60.7.70/ 3002
A8	Protocol Configuration Register 4 (FR_PCR4)	16	R/W	0000h	60.7.71/ 3002
AA	Protocol Configuration Register 5 (FR_PCR5)	16	R/W	0000h	60.7.72/ 3003
AC	Protocol Configuration Register 6 (FR_PCR6)	16	R/W	0000h	60.7.73/ 3003
AE	Protocol Configuration Register 7 (FR_PCR7)	16	R/W	0000h	60.7.74/ 3004
B0	Protocol Configuration Register 8 (FR_PCR8)	16	R/W	0000h	60.7.75/ 3004
B2	Protocol Configuration Register 9 (FR_PCR9)	16	R/W	0000h	60.7.76/ 3005

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B4	Protocol Configuration Register 10 (FR_PCR10)	16	R/W	0000h	60.7.77/ 3006
B6	Protocol Configuration Register 11 (FR_PCR11)	16	R/W	0000h	60.7.78/ 3006
B8	Protocol Configuration Register 12 (FR_PCR12)	16	R/W	0000h	60.7.79/ 3007
BA	Protocol Configuration Register 13 (FR_PCR13)	16	R/W	0000h	60.7.80/ 3008
BC	Protocol Configuration Register 14 (FR_PCR14)	16	R/W	0000h	60.7.81/ 3008
BE	Protocol Configuration Register 15 (FR_PCR15)	16	R/W	0000h	60.7.82/ 3009
C0	Protocol Configuration Register 16 (FR_PCR16)	16	R/W	0000h	60.7.83/ 3009
C2	Protocol Configuration Register 17 (FR_PCR17)	16	R/W	0000h	60.7.84/ 3010
C4	Protocol Configuration Register 18 (FR_PCR18)	16	R/W	0000h	60.7.85/ 3010
C6	Protocol Configuration Register 19 (FR_PCR19)	16	R/W	0000h	60.7.86/ 3011
C8	Protocol Configuration Register 20 (FR_PCR20)	16	R/W	0000h	60.7.87/ 3011
CA	Protocol Configuration Register 21 (FR_PCR21)	16	R/W	0000h	60.7.88/ 3012
CC	Protocol Configuration Register 22 (FR_PCR22)	16	R/W	0000h	60.7.89/ 3012
CE	Protocol Configuration Register 23 (FR_PCR23)	16	R/W	0000h	60.7.90/ 3013
D0	Protocol Configuration Register 24 (FR_PCR24)	16	R/W	0000h	60.7.91/ 3013
D2	Protocol Configuration Register 25 (FR_PCR25)	16	R/W	0000h	60.7.92/ 3014
D4	Protocol Configuration Register 26 (FR_PCR26)	16	R/W	0000h	60.7.93/ 3014
D6	Protocol Configuration Register 27 (FR_PCR27)	16	R/W	0000h	60.7.94/ 3015
D8	Protocol Configuration Register 28 (FR_PCR28)	16	R/W	0000h	60.7.95/ 3016
DA	Protocol Configuration Register 29 (FR_PCR29)	16	R/W	0000h	60.7.96/ 3016
DC	Protocol Configuration Register 30 (FR_PCR30)	16	R/W	0000h	60.7.97/ 3017
E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	16	R/W	0000h	60.7.98/ 3017

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	16	R/W	0000h	60.7.99/ 3018
EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	16	R/W	0000h	60.7.100/ 3018
EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	16	R/W	0000h	60.7.101/ 3019
EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	16	R/W	0000h	60.7.102/ 3019
F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	16	R/W	0000h	60.7.103/ 3021
F2	ECC Error Report and Injection Control Register (FR_EERICR)	16	R/W	0000h	60.7.104/ 3023
F4	ECC Error Report Address Register (FR_EERAR)	16	R	7000h	60.7.105/ 3024
F6	ECC Error Report Data Register (FR_EERDR)	16	R	0000h	60.7.106/ 3025
F8	ECC Error Report Code Register (FR_EERCR)	16	R	0000h	60.7.107/ 3026
FA	ECC Error Injection Address Register (FR_EEIAR)	16	R/W	0000h	60.7.108/ 3027
FC	ECC Error Injection Data Register (FR_EEIDR)	16	R/W	0000h	60.7.109/ 3027
FE	ECC Error Injection Code Register (FR_EEICR)	16	R/W	0000h	60.7.110/ 3028
800	Message Buffer Configuration, Control, Status Register (FR_MBCCSR0)	16	R/W	0000h	60.7.111/ 3028
802	Message Buffer Cycle Counter Filter Register (FR_MBCCFR0)	16	R/W	See section	60.7.112/ 3030
804	Message Buffer Frame ID Register (FR_MBFIDR0)	16	R/W	See section	60.7.113/ 3032
806	Message Buffer Index Register (FR_MBIDXR0)	16	R/W	See section	60.7.114/ 3032
808	Message Buffer Configuration, Control, Status Register (FR_MBCCSR1)	16	R/W	0000h	60.7.111/ 3028
80A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR1)	16	R/W	See section	60.7.112/ 3030
80C	Message Buffer Frame ID Register (FR_MBFIDR1)	16	R/W	See section	60.7.113/ 3032
80E	Message Buffer Index Register (FR_MBIDXR1)	16	R/W	See section	60.7.114/ 3032
810	Message Buffer Configuration, Control, Status Register (FR_MBCCSR2)	16	R/W	0000h	60.7.111/ 3028
812	Message Buffer Cycle Counter Filter Register (FR_MBCCFR2)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
814	Message Buffer Frame ID Register (FR_MBFIDR2)	16	R/W	See section	60.7.113/ 3032
816	Message Buffer Index Register (FR_MBIDXR2)	16	R/W	See section	60.7.114/ 3032
818	Message Buffer Configuration, Control, Status Register (FR_MBCCSR3)	16	R/W	0000h	60.7.111/ 3028
81A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR3)	16	R/W	See section	60.7.112/ 3030
81C	Message Buffer Frame ID Register (FR_MBFIDR3)	16	R/W	See section	60.7.113/ 3032
81E	Message Buffer Index Register (FR_MBIDXR3)	16	R/W	See section	60.7.114/ 3032
820	Message Buffer Configuration, Control, Status Register (FR_MBCCSR4)	16	R/W	0000h	60.7.111/ 3028
822	Message Buffer Cycle Counter Filter Register (FR_MBCCFR4)	16	R/W	See section	60.7.112/ 3030
824	Message Buffer Frame ID Register (FR_MBFIDR4)	16	R/W	See section	60.7.113/ 3032
826	Message Buffer Index Register (FR_MBIDXR4)	16	R/W	See section	60.7.114/ 3032
828	Message Buffer Configuration, Control, Status Register (FR_MBCCSR5)	16	R/W	0000h	60.7.111/ 3028
82A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR5)	16	R/W	See section	60.7.112/ 3030
82C	Message Buffer Frame ID Register (FR_MBFIDR5)	16	R/W	See section	60.7.113/ 3032
82E	Message Buffer Index Register (FR_MBIDXR5)	16	R/W	See section	60.7.114/ 3032
830	Message Buffer Configuration, Control, Status Register (FR_MBCCSR6)	16	R/W	0000h	60.7.111/ 3028
832	Message Buffer Cycle Counter Filter Register (FR_MBCCFR6)	16	R/W	See section	60.7.112/ 3030
834	Message Buffer Frame ID Register (FR_MBFIDR6)	16	R/W	See section	60.7.113/ 3032
836	Message Buffer Index Register (FR_MBIDXR6)	16	R/W	See section	60.7.114/ 3032
838	Message Buffer Configuration, Control, Status Register (FR_MBCCSR7)	16	R/W	0000h	60.7.111/ 3028
83A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR7)	16	R/W	See section	60.7.112/ 3030
83C	Message Buffer Frame ID Register (FR_MBFIDR7)	16	R/W	See section	60.7.113/ 3032
83E	Message Buffer Index Register (FR_MBIDXR7)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
840	Message Buffer Configuration, Control, Status Register (FR_MBCCSR8)	16	R/W	0000h	60.7.111/ 3028
842	Message Buffer Cycle Counter Filter Register (FR_MBCCFR8)	16	R/W	See section	60.7.112/ 3030
844	Message Buffer Frame ID Register (FR_MBFIDR8)	16	R/W	See section	60.7.113/ 3032
846	Message Buffer Index Register (FR_MBIDXR8)	16	R/W	See section	60.7.114/ 3032
848	Message Buffer Configuration, Control, Status Register (FR_MBCCSR9)	16	R/W	0000h	60.7.111/ 3028
84A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR9)	16	R/W	See section	60.7.112/ 3030
84C	Message Buffer Frame ID Register (FR_MBFIDR9)	16	R/W	See section	60.7.113/ 3032
84E	Message Buffer Index Register (FR_MBIDXR9)	16	R/W	See section	60.7.114/ 3032
850	Message Buffer Configuration, Control, Status Register (FR_MBCCSR10)	16	R/W	0000h	60.7.111/ 3028
852	Message Buffer Cycle Counter Filter Register (FR_MBCCFR10)	16	R/W	See section	60.7.112/ 3030
854	Message Buffer Frame ID Register (FR_MBFIDR10)	16	R/W	See section	60.7.113/ 3032
856	Message Buffer Index Register (FR_MBIDXR10)	16	R/W	See section	60.7.114/ 3032
858	Message Buffer Configuration, Control, Status Register (FR_MBCCSR11)	16	R/W	0000h	60.7.111/ 3028
85A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR11)	16	R/W	See section	60.7.112/ 3030
85C	Message Buffer Frame ID Register (FR_MBFIDR11)	16	R/W	See section	60.7.113/ 3032
85E	Message Buffer Index Register (FR_MBIDXR11)	16	R/W	See section	60.7.114/ 3032
860	Message Buffer Configuration, Control, Status Register (FR_MBCCSR12)	16	R/W	0000h	60.7.111/ 3028
862	Message Buffer Cycle Counter Filter Register (FR_MBCCFR12)	16	R/W	See section	60.7.112/ 3030
864	Message Buffer Frame ID Register (FR_MBFIDR12)	16	R/W	See section	60.7.113/ 3032
866	Message Buffer Index Register (FR_MBIDXR12)	16	R/W	See section	60.7.114/ 3032
868	Message Buffer Configuration, Control, Status Register (FR_MBCCSR13)	16	R/W	0000h	60.7.111/ 3028
86A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR13)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
86C	Message Buffer Frame ID Register (FR_MBFIDR13)	16	R/W	See section	60.7.113/ 3032
86E	Message Buffer Index Register (FR_MBIDXR13)	16	R/W	See section	60.7.114/ 3032
870	Message Buffer Configuration, Control, Status Register (FR_MBCCSR14)	16	R/W	0000h	60.7.111/ 3028
872	Message Buffer Cycle Counter Filter Register (FR_MBCCFR14)	16	R/W	See section	60.7.112/ 3030
874	Message Buffer Frame ID Register (FR_MBFIDR14)	16	R/W	See section	60.7.113/ 3032
876	Message Buffer Index Register (FR_MBIDXR14)	16	R/W	See section	60.7.114/ 3032
878	Message Buffer Configuration, Control, Status Register (FR_MBCCSR15)	16	R/W	0000h	60.7.111/ 3028
87A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR15)	16	R/W	See section	60.7.112/ 3030
87C	Message Buffer Frame ID Register (FR_MBFIDR15)	16	R/W	See section	60.7.113/ 3032
87E	Message Buffer Index Register (FR_MBIDXR15)	16	R/W	See section	60.7.114/ 3032
880	Message Buffer Configuration, Control, Status Register (FR_MBCCSR16)	16	R/W	0000h	60.7.111/ 3028
882	Message Buffer Cycle Counter Filter Register (FR_MBCCFR16)	16	R/W	See section	60.7.112/ 3030
884	Message Buffer Frame ID Register (FR_MBFIDR16)	16	R/W	See section	60.7.113/ 3032
886	Message Buffer Index Register (FR_MBIDXR16)	16	R/W	See section	60.7.114/ 3032
888	Message Buffer Configuration, Control, Status Register (FR_MBCCSR17)	16	R/W	0000h	60.7.111/ 3028
88A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR17)	16	R/W	See section	60.7.112/ 3030
88C	Message Buffer Frame ID Register (FR_MBFIDR17)	16	R/W	See section	60.7.113/ 3032
88E	Message Buffer Index Register (FR_MBIDXR17)	16	R/W	See section	60.7.114/ 3032
890	Message Buffer Configuration, Control, Status Register (FR_MBCCSR18)	16	R/W	0000h	60.7.111/ 3028
892	Message Buffer Cycle Counter Filter Register (FR_MBCCFR18)	16	R/W	See section	60.7.112/ 3030
894	Message Buffer Frame ID Register (FR_MBFIDR18)	16	R/W	See section	60.7.113/ 3032
896	Message Buffer Index Register (FR_MBIDXR18)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
898	Message Buffer Configuration, Control, Status Register (FR_MBCCSR19)	16	R/W	0000h	60.7.111/ 3028
89A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR19)	16	R/W	See section	60.7.112/ 3030
89C	Message Buffer Frame ID Register (FR_MBFIDR19)	16	R/W	See section	60.7.113/ 3032
89E	Message Buffer Index Register (FR_MBIDXR19)	16	R/W	See section	60.7.114/ 3032
8A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR20)	16	R/W	0000h	60.7.111/ 3028
8A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR20)	16	R/W	See section	60.7.112/ 3030
8A4	Message Buffer Frame ID Register (FR_MBFIDR20)	16	R/W	See section	60.7.113/ 3032
8A6	Message Buffer Index Register (FR_MBIDXR20)	16	R/W	See section	60.7.114/ 3032
8A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR21)	16	R/W	0000h	60.7.111/ 3028
8AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR21)	16	R/W	See section	60.7.112/ 3030
8AC	Message Buffer Frame ID Register (FR_MBFIDR21)	16	R/W	See section	60.7.113/ 3032
8AE	Message Buffer Index Register (FR_MBIDXR21)	16	R/W	See section	60.7.114/ 3032
8B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR22)	16	R/W	0000h	60.7.111/ 3028
8B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR22)	16	R/W	See section	60.7.112/ 3030
8B4	Message Buffer Frame ID Register (FR_MBFIDR22)	16	R/W	See section	60.7.113/ 3032
8B6	Message Buffer Index Register (FR_MBIDXR22)	16	R/W	See section	60.7.114/ 3032
8B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR23)	16	R/W	0000h	60.7.111/ 3028
8BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR23)	16	R/W	See section	60.7.112/ 3030
8BC	Message Buffer Frame ID Register (FR_MBFIDR23)	16	R/W	See section	60.7.113/ 3032
8BE	Message Buffer Index Register (FR_MBIDXR23)	16	R/W	See section	60.7.114/ 3032
8C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR24)	16	R/W	0000h	60.7.111/ 3028
8C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR24)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C4	Message Buffer Frame ID Register (FR_MBFIDR24)	16	R/W	See section	60.7.113/ 3032
8C6	Message Buffer Index Register (FR_MBIDXR24)	16	R/W	See section	60.7.114/ 3032
8C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR25)	16	R/W	0000h	60.7.111/ 3028
8CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR25)	16	R/W	See section	60.7.112/ 3030
8CC	Message Buffer Frame ID Register (FR_MBFIDR25)	16	R/W	See section	60.7.113/ 3032
8CE	Message Buffer Index Register (FR_MBIDXR25)	16	R/W	See section	60.7.114/ 3032
8D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR26)	16	R/W	0000h	60.7.111/ 3028
8D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR26)	16	R/W	See section	60.7.112/ 3030
8D4	Message Buffer Frame ID Register (FR_MBFIDR26)	16	R/W	See section	60.7.113/ 3032
8D6	Message Buffer Index Register (FR_MBIDXR26)	16	R/W	See section	60.7.114/ 3032
8D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR27)	16	R/W	0000h	60.7.111/ 3028
8DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR27)	16	R/W	See section	60.7.112/ 3030
8DC	Message Buffer Frame ID Register (FR_MBFIDR27)	16	R/W	See section	60.7.113/ 3032
8DE	Message Buffer Index Register (FR_MBIDXR27)	16	R/W	See section	60.7.114/ 3032
8E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR28)	16	R/W	0000h	60.7.111/ 3028
8E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR28)	16	R/W	See section	60.7.112/ 3030
8E4	Message Buffer Frame ID Register (FR_MBFIDR28)	16	R/W	See section	60.7.113/ 3032
8E6	Message Buffer Index Register (FR_MBIDXR28)	16	R/W	See section	60.7.114/ 3032
8E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR29)	16	R/W	0000h	60.7.111/ 3028
8EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR29)	16	R/W	See section	60.7.112/ 3030
8EC	Message Buffer Frame ID Register (FR_MBFIDR29)	16	R/W	See section	60.7.113/ 3032
8EE	Message Buffer Index Register (FR_MBIDXR29)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR30)	16	R/W	0000h	60.7.111/ 3028
8F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR30)	16	R/W	See section	60.7.112/ 3030
8F4	Message Buffer Frame ID Register (FR_MBFIDR30)	16	R/W	See section	60.7.113/ 3032
8F6	Message Buffer Index Register (FR_MBIDXR30)	16	R/W	See section	60.7.114/ 3032
8F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR31)	16	R/W	0000h	60.7.111/ 3028
8FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR31)	16	R/W	See section	60.7.112/ 3030
8FC	Message Buffer Frame ID Register (FR_MBFIDR31)	16	R/W	See section	60.7.113/ 3032
8FE	Message Buffer Index Register (FR_MBIDXR31)	16	R/W	See section	60.7.114/ 3032
900	Message Buffer Configuration, Control, Status Register (FR_MBCCSR32)	16	R/W	0000h	60.7.111/ 3028
902	Message Buffer Cycle Counter Filter Register (FR_MBCCFR32)	16	R/W	See section	60.7.112/ 3030
904	Message Buffer Frame ID Register (FR_MBFIDR32)	16	R/W	See section	60.7.113/ 3032
906	Message Buffer Index Register (FR_MBIDXR32)	16	R/W	See section	60.7.114/ 3032
908	Message Buffer Configuration, Control, Status Register (FR_MBCCSR33)	16	R/W	0000h	60.7.111/ 3028
90A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR33)	16	R/W	See section	60.7.112/ 3030
90C	Message Buffer Frame ID Register (FR_MBFIDR33)	16	R/W	See section	60.7.113/ 3032
90E	Message Buffer Index Register (FR_MBIDXR33)	16	R/W	See section	60.7.114/ 3032
910	Message Buffer Configuration, Control, Status Register (FR_MBCCSR34)	16	R/W	0000h	60.7.111/ 3028
912	Message Buffer Cycle Counter Filter Register (FR_MBCCFR34)	16	R/W	See section	60.7.112/ 3030
914	Message Buffer Frame ID Register (FR_MBFIDR34)	16	R/W	See section	60.7.113/ 3032
916	Message Buffer Index Register (FR_MBIDXR34)	16	R/W	See section	60.7.114/ 3032
918	Message Buffer Configuration, Control, Status Register (FR_MBCCSR35)	16	R/W	0000h	60.7.111/ 3028
91A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR35)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
91C	Message Buffer Frame ID Register (FR_MBFIDR35)	16	R/W	See section	60.7.113/ 3032
91E	Message Buffer Index Register (FR_MBIDXR35)	16	R/W	See section	60.7.114/ 3032
920	Message Buffer Configuration, Control, Status Register (FR_MBCCSR36)	16	R/W	0000h	60.7.111/ 3028
922	Message Buffer Cycle Counter Filter Register (FR_MBCCFR36)	16	R/W	See section	60.7.112/ 3030
924	Message Buffer Frame ID Register (FR_MBFIDR36)	16	R/W	See section	60.7.113/ 3032
926	Message Buffer Index Register (FR_MBIDXR36)	16	R/W	See section	60.7.114/ 3032
928	Message Buffer Configuration, Control, Status Register (FR_MBCCSR37)	16	R/W	0000h	60.7.111/ 3028
92A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR37)	16	R/W	See section	60.7.112/ 3030
92C	Message Buffer Frame ID Register (FR_MBFIDR37)	16	R/W	See section	60.7.113/ 3032
92E	Message Buffer Index Register (FR_MBIDXR37)	16	R/W	See section	60.7.114/ 3032
930	Message Buffer Configuration, Control, Status Register (FR_MBCCSR38)	16	R/W	0000h	60.7.111/ 3028
932	Message Buffer Cycle Counter Filter Register (FR_MBCCFR38)	16	R/W	See section	60.7.112/ 3030
934	Message Buffer Frame ID Register (FR_MBFIDR38)	16	R/W	See section	60.7.113/ 3032
936	Message Buffer Index Register (FR_MBIDXR38)	16	R/W	See section	60.7.114/ 3032
938	Message Buffer Configuration, Control, Status Register (FR_MBCCSR39)	16	R/W	0000h	60.7.111/ 3028
93A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR39)	16	R/W	See section	60.7.112/ 3030
93C	Message Buffer Frame ID Register (FR_MBFIDR39)	16	R/W	See section	60.7.113/ 3032
93E	Message Buffer Index Register (FR_MBIDXR39)	16	R/W	See section	60.7.114/ 3032
940	Message Buffer Configuration, Control, Status Register (FR_MBCCSR40)	16	R/W	0000h	60.7.111/ 3028
942	Message Buffer Cycle Counter Filter Register (FR_MBCCFR40)	16	R/W	See section	60.7.112/ 3030
944	Message Buffer Frame ID Register (FR_MBFIDR40)	16	R/W	See section	60.7.113/ 3032
946	Message Buffer Index Register (FR_MBIDXR40)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
948	Message Buffer Configuration, Control, Status Register (FR_MBCCSR41)	16	R/W	0000h	60.7.111/ 3028
94A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR41)	16	R/W	See section	60.7.112/ 3030
94C	Message Buffer Frame ID Register (FR_MBFIDR41)	16	R/W	See section	60.7.113/ 3032
94E	Message Buffer Index Register (FR_MBIDXR41)	16	R/W	See section	60.7.114/ 3032
950	Message Buffer Configuration, Control, Status Register (FR_MBCCSR42)	16	R/W	0000h	60.7.111/ 3028
952	Message Buffer Cycle Counter Filter Register (FR_MBCCFR42)	16	R/W	See section	60.7.112/ 3030
954	Message Buffer Frame ID Register (FR_MBFIDR42)	16	R/W	See section	60.7.113/ 3032
956	Message Buffer Index Register (FR_MBIDXR42)	16	R/W	See section	60.7.114/ 3032
958	Message Buffer Configuration, Control, Status Register (FR_MBCCSR43)	16	R/W	0000h	60.7.111/ 3028
95A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR43)	16	R/W	See section	60.7.112/ 3030
95C	Message Buffer Frame ID Register (FR_MBFIDR43)	16	R/W	See section	60.7.113/ 3032
95E	Message Buffer Index Register (FR_MBIDXR43)	16	R/W	See section	60.7.114/ 3032
960	Message Buffer Configuration, Control, Status Register (FR_MBCCSR44)	16	R/W	0000h	60.7.111/ 3028
962	Message Buffer Cycle Counter Filter Register (FR_MBCCFR44)	16	R/W	See section	60.7.112/ 3030
964	Message Buffer Frame ID Register (FR_MBFIDR44)	16	R/W	See section	60.7.113/ 3032
966	Message Buffer Index Register (FR_MBIDXR44)	16	R/W	See section	60.7.114/ 3032
968	Message Buffer Configuration, Control, Status Register (FR_MBCCSR45)	16	R/W	0000h	60.7.111/ 3028
96A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR45)	16	R/W	See section	60.7.112/ 3030
96C	Message Buffer Frame ID Register (FR_MBFIDR45)	16	R/W	See section	60.7.113/ 3032
96E	Message Buffer Index Register (FR_MBIDXR45)	16	R/W	See section	60.7.114/ 3032
970	Message Buffer Configuration, Control, Status Register (FR_MBCCSR46)	16	R/W	0000h	60.7.111/ 3028
972	Message Buffer Cycle Counter Filter Register (FR_MBCCFR46)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
974	Message Buffer Frame ID Register (FR_MBFIDR46)	16	R/W	See section	60.7.113/ 3032
976	Message Buffer Index Register (FR_MBIDXR46)	16	R/W	See section	60.7.114/ 3032
978	Message Buffer Configuration, Control, Status Register (FR_MBCCSR47)	16	R/W	0000h	60.7.111/ 3028
97A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR47)	16	R/W	See section	60.7.112/ 3030
97C	Message Buffer Frame ID Register (FR_MBFIDR47)	16	R/W	See section	60.7.113/ 3032
97E	Message Buffer Index Register (FR_MBIDXR47)	16	R/W	See section	60.7.114/ 3032
980	Message Buffer Configuration, Control, Status Register (FR_MBCCSR48)	16	R/W	0000h	60.7.111/ 3028
982	Message Buffer Cycle Counter Filter Register (FR_MBCCFR48)	16	R/W	See section	60.7.112/ 3030
984	Message Buffer Frame ID Register (FR_MBFIDR48)	16	R/W	See section	60.7.113/ 3032
986	Message Buffer Index Register (FR_MBIDXR48)	16	R/W	See section	60.7.114/ 3032
988	Message Buffer Configuration, Control, Status Register (FR_MBCCSR49)	16	R/W	0000h	60.7.111/ 3028
98A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR49)	16	R/W	See section	60.7.112/ 3030
98C	Message Buffer Frame ID Register (FR_MBFIDR49)	16	R/W	See section	60.7.113/ 3032
98E	Message Buffer Index Register (FR_MBIDXR49)	16	R/W	See section	60.7.114/ 3032
990	Message Buffer Configuration, Control, Status Register (FR_MBCCSR50)	16	R/W	0000h	60.7.111/ 3028
992	Message Buffer Cycle Counter Filter Register (FR_MBCCFR50)	16	R/W	See section	60.7.112/ 3030
994	Message Buffer Frame ID Register (FR_MBFIDR50)	16	R/W	See section	60.7.113/ 3032
996	Message Buffer Index Register (FR_MBIDXR50)	16	R/W	See section	60.7.114/ 3032
998	Message Buffer Configuration, Control, Status Register (FR_MBCCSR51)	16	R/W	0000h	60.7.111/ 3028
99A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR51)	16	R/W	See section	60.7.112/ 3030
99C	Message Buffer Frame ID Register (FR_MBFIDR51)	16	R/W	See section	60.7.113/ 3032
99E	Message Buffer Index Register (FR_MBIDXR51)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR52)	16	R/W	0000h	60.7.111/ 3028
9A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR52)	16	R/W	See section	60.7.112/ 3030
9A4	Message Buffer Frame ID Register (FR_MBFIDR52)	16	R/W	See section	60.7.113/ 3032
9A6	Message Buffer Index Register (FR_MBIDXR52)	16	R/W	See section	60.7.114/ 3032
9A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR53)	16	R/W	0000h	60.7.111/ 3028
9AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR53)	16	R/W	See section	60.7.112/ 3030
9AC	Message Buffer Frame ID Register (FR_MBFIDR53)	16	R/W	See section	60.7.113/ 3032
9AE	Message Buffer Index Register (FR_MBIDXR53)	16	R/W	See section	60.7.114/ 3032
9B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR54)	16	R/W	0000h	60.7.111/ 3028
9B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR54)	16	R/W	See section	60.7.112/ 3030
9B4	Message Buffer Frame ID Register (FR_MBFIDR54)	16	R/W	See section	60.7.113/ 3032
9B6	Message Buffer Index Register (FR_MBIDXR54)	16	R/W	See section	60.7.114/ 3032
9B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR55)	16	R/W	0000h	60.7.111/ 3028
9BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR55)	16	R/W	See section	60.7.112/ 3030
9BC	Message Buffer Frame ID Register (FR_MBFIDR55)	16	R/W	See section	60.7.113/ 3032
9BE	Message Buffer Index Register (FR_MBIDXR55)	16	R/W	See section	60.7.114/ 3032
9C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR56)	16	R/W	0000h	60.7.111/ 3028
9C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR56)	16	R/W	See section	60.7.112/ 3030
9C4	Message Buffer Frame ID Register (FR_MBFIDR56)	16	R/W	See section	60.7.113/ 3032
9C6	Message Buffer Index Register (FR_MBIDXR56)	16	R/W	See section	60.7.114/ 3032
9C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR57)	16	R/W	0000h	60.7.111/ 3028
9CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR57)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9CC	Message Buffer Frame ID Register (FR_MBFIDR57)	16	R/W	See section	60.7.113/ 3032
9CE	Message Buffer Index Register (FR_MBIDXR57)	16	R/W	See section	60.7.114/ 3032
9D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR58)	16	R/W	0000h	60.7.111/ 3028
9D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR58)	16	R/W	See section	60.7.112/ 3030
9D4	Message Buffer Frame ID Register (FR_MBFIDR58)	16	R/W	See section	60.7.113/ 3032
9D6	Message Buffer Index Register (FR_MBIDXR58)	16	R/W	See section	60.7.114/ 3032
9D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR59)	16	R/W	0000h	60.7.111/ 3028
9DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR59)	16	R/W	See section	60.7.112/ 3030
9DC	Message Buffer Frame ID Register (FR_MBFIDR59)	16	R/W	See section	60.7.113/ 3032
9DE	Message Buffer Index Register (FR_MBIDXR59)	16	R/W	See section	60.7.114/ 3032
9E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR60)	16	R/W	0000h	60.7.111/ 3028
9E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR60)	16	R/W	See section	60.7.112/ 3030
9E4	Message Buffer Frame ID Register (FR_MBFIDR60)	16	R/W	See section	60.7.113/ 3032
9E6	Message Buffer Index Register (FR_MBIDXR60)	16	R/W	See section	60.7.114/ 3032
9E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR61)	16	R/W	0000h	60.7.111/ 3028
9EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR61)	16	R/W	See section	60.7.112/ 3030
9EC	Message Buffer Frame ID Register (FR_MBFIDR61)	16	R/W	See section	60.7.113/ 3032
9EE	Message Buffer Index Register (FR_MBIDXR61)	16	R/W	See section	60.7.114/ 3032
9F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR62)	16	R/W	0000h	60.7.111/ 3028
9F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR62)	16	R/W	See section	60.7.112/ 3030
9F4	Message Buffer Frame ID Register (FR_MBFIDR62)	16	R/W	See section	60.7.113/ 3032
9F6	Message Buffer Index Register (FR_MBIDXR62)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR63)	16	R/W	0000h	60.7.111/ 3028
9FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR63)	16	R/W	See section	60.7.112/ 3030
9FC	Message Buffer Frame ID Register (FR_MBFIDR63)	16	R/W	See section	60.7.113/ 3032
9FE	Message Buffer Index Register (FR_MBIDXR63)	16	R/W	See section	60.7.114/ 3032
A00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR64)	16	R/W	0000h	60.7.111/ 3028
A02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR64)	16	R/W	See section	60.7.112/ 3030
A04	Message Buffer Frame ID Register (FR_MBFIDR64)	16	R/W	See section	60.7.113/ 3032
A06	Message Buffer Index Register (FR_MBIDXR64)	16	R/W	See section	60.7.114/ 3032
A08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR65)	16	R/W	0000h	60.7.111/ 3028
A0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR65)	16	R/W	See section	60.7.112/ 3030
A0C	Message Buffer Frame ID Register (FR_MBFIDR65)	16	R/W	See section	60.7.113/ 3032
A0E	Message Buffer Index Register (FR_MBIDXR65)	16	R/W	See section	60.7.114/ 3032
A10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR66)	16	R/W	0000h	60.7.111/ 3028
A12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR66)	16	R/W	See section	60.7.112/ 3030
A14	Message Buffer Frame ID Register (FR_MBFIDR66)	16	R/W	See section	60.7.113/ 3032
A16	Message Buffer Index Register (FR_MBIDXR66)	16	R/W	See section	60.7.114/ 3032
A18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR67)	16	R/W	0000h	60.7.111/ 3028
A1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR67)	16	R/W	See section	60.7.112/ 3030
A1C	Message Buffer Frame ID Register (FR_MBFIDR67)	16	R/W	See section	60.7.113/ 3032
A1E	Message Buffer Index Register (FR_MBIDXR67)	16	R/W	See section	60.7.114/ 3032
A20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR68)	16	R/W	0000h	60.7.111/ 3028
A22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR68)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A24	Message Buffer Frame ID Register (FR_MBFIDR68)	16	R/W	See section	60.7.113/ 3032
A26	Message Buffer Index Register (FR_MBIDXR68)	16	R/W	See section	60.7.114/ 3032
A28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR69)	16	R/W	0000h	60.7.111/ 3028
A2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR69)	16	R/W	See section	60.7.112/ 3030
A2C	Message Buffer Frame ID Register (FR_MBFIDR69)	16	R/W	See section	60.7.113/ 3032
A2E	Message Buffer Index Register (FR_MBIDXR69)	16	R/W	See section	60.7.114/ 3032
A30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR70)	16	R/W	0000h	60.7.111/ 3028
A32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR70)	16	R/W	See section	60.7.112/ 3030
A34	Message Buffer Frame ID Register (FR_MBFIDR70)	16	R/W	See section	60.7.113/ 3032
A36	Message Buffer Index Register (FR_MBIDXR70)	16	R/W	See section	60.7.114/ 3032
A38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR71)	16	R/W	0000h	60.7.111/ 3028
A3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR71)	16	R/W	See section	60.7.112/ 3030
A3C	Message Buffer Frame ID Register (FR_MBFIDR71)	16	R/W	See section	60.7.113/ 3032
A3E	Message Buffer Index Register (FR_MBIDXR71)	16	R/W	See section	60.7.114/ 3032
A40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR72)	16	R/W	0000h	60.7.111/ 3028
A42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR72)	16	R/W	See section	60.7.112/ 3030
A44	Message Buffer Frame ID Register (FR_MBFIDR72)	16	R/W	See section	60.7.113/ 3032
A46	Message Buffer Index Register (FR_MBIDXR72)	16	R/W	See section	60.7.114/ 3032
A48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR73)	16	R/W	0000h	60.7.111/ 3028
A4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR73)	16	R/W	See section	60.7.112/ 3030
A4C	Message Buffer Frame ID Register (FR_MBFIDR73)	16	R/W	See section	60.7.113/ 3032
A4E	Message Buffer Index Register (FR_MBIDXR73)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR74)	16	R/W	0000h	60.7.111/ 3028
A52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR74)	16	R/W	See section	60.7.112/ 3030
A54	Message Buffer Frame ID Register (FR_MBFIDR74)	16	R/W	See section	60.7.113/ 3032
A56	Message Buffer Index Register (FR_MBIDXR74)	16	R/W	See section	60.7.114/ 3032
A58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR75)	16	R/W	0000h	60.7.111/ 3028
A5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR75)	16	R/W	See section	60.7.112/ 3030
A5C	Message Buffer Frame ID Register (FR_MBFIDR75)	16	R/W	See section	60.7.113/ 3032
A5E	Message Buffer Index Register (FR_MBIDXR75)	16	R/W	See section	60.7.114/ 3032
A60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR76)	16	R/W	0000h	60.7.111/ 3028
A62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR76)	16	R/W	See section	60.7.112/ 3030
A64	Message Buffer Frame ID Register (FR_MBFIDR76)	16	R/W	See section	60.7.113/ 3032
A66	Message Buffer Index Register (FR_MBIDXR76)	16	R/W	See section	60.7.114/ 3032
A68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR77)	16	R/W	0000h	60.7.111/ 3028
A6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR77)	16	R/W	See section	60.7.112/ 3030
A6C	Message Buffer Frame ID Register (FR_MBFIDR77)	16	R/W	See section	60.7.113/ 3032
A6E	Message Buffer Index Register (FR_MBIDXR77)	16	R/W	See section	60.7.114/ 3032
A70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR78)	16	R/W	0000h	60.7.111/ 3028
A72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR78)	16	R/W	See section	60.7.112/ 3030
A74	Message Buffer Frame ID Register (FR_MBFIDR78)	16	R/W	See section	60.7.113/ 3032
A76	Message Buffer Index Register (FR_MBIDXR78)	16	R/W	See section	60.7.114/ 3032
A78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR79)	16	R/W	0000h	60.7.111/ 3028
A7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR79)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A7C	Message Buffer Frame ID Register (FR_MBFIDR79)	16	R/W	See section	60.7.113/ 3032
A7E	Message Buffer Index Register (FR_MBIDXR79)	16	R/W	See section	60.7.114/ 3032
A80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR80)	16	R/W	0000h	60.7.111/ 3028
A82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR80)	16	R/W	See section	60.7.112/ 3030
A84	Message Buffer Frame ID Register (FR_MBFIDR80)	16	R/W	See section	60.7.113/ 3032
A86	Message Buffer Index Register (FR_MBIDXR80)	16	R/W	See section	60.7.114/ 3032
A88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR81)	16	R/W	0000h	60.7.111/ 3028
A8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR81)	16	R/W	See section	60.7.112/ 3030
A8C	Message Buffer Frame ID Register (FR_MBFIDR81)	16	R/W	See section	60.7.113/ 3032
A8E	Message Buffer Index Register (FR_MBIDXR81)	16	R/W	See section	60.7.114/ 3032
A90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR82)	16	R/W	0000h	60.7.111/ 3028
A92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR82)	16	R/W	See section	60.7.112/ 3030
A94	Message Buffer Frame ID Register (FR_MBFIDR82)	16	R/W	See section	60.7.113/ 3032
A96	Message Buffer Index Register (FR_MBIDXR82)	16	R/W	See section	60.7.114/ 3032
A98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR83)	16	R/W	0000h	60.7.111/ 3028
A9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR83)	16	R/W	See section	60.7.112/ 3030
A9C	Message Buffer Frame ID Register (FR_MBFIDR83)	16	R/W	See section	60.7.113/ 3032
A9E	Message Buffer Index Register (FR_MBIDXR83)	16	R/W	See section	60.7.114/ 3032
AA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR84)	16	R/W	0000h	60.7.111/ 3028
AA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR84)	16	R/W	See section	60.7.112/ 3030
AA4	Message Buffer Frame ID Register (FR_MBFIDR84)	16	R/W	See section	60.7.113/ 3032
AA6	Message Buffer Index Register (FR_MBIDXR84)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR85)	16	R/W	0000h	60.7.111/ 3028
AAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR85)	16	R/W	See section	60.7.112/ 3030
AAC	Message Buffer Frame ID Register (FR_MBFIDR85)	16	R/W	See section	60.7.113/ 3032
AAE	Message Buffer Index Register (FR_MBIDXR85)	16	R/W	See section	60.7.114/ 3032
AB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR86)	16	R/W	0000h	60.7.111/ 3028
AB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR86)	16	R/W	See section	60.7.112/ 3030
AB4	Message Buffer Frame ID Register (FR_MBFIDR86)	16	R/W	See section	60.7.113/ 3032
AB6	Message Buffer Index Register (FR_MBIDXR86)	16	R/W	See section	60.7.114/ 3032
AB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR87)	16	R/W	0000h	60.7.111/ 3028
ABA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR87)	16	R/W	See section	60.7.112/ 3030
ABC	Message Buffer Frame ID Register (FR_MBFIDR87)	16	R/W	See section	60.7.113/ 3032
ABE	Message Buffer Index Register (FR_MBIDXR87)	16	R/W	See section	60.7.114/ 3032
AC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR88)	16	R/W	0000h	60.7.111/ 3028
AC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR88)	16	R/W	See section	60.7.112/ 3030
AC4	Message Buffer Frame ID Register (FR_MBFIDR88)	16	R/W	See section	60.7.113/ 3032
AC6	Message Buffer Index Register (FR_MBIDXR88)	16	R/W	See section	60.7.114/ 3032
AC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR89)	16	R/W	0000h	60.7.111/ 3028
ACA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR89)	16	R/W	See section	60.7.112/ 3030
ACC	Message Buffer Frame ID Register (FR_MBFIDR89)	16	R/W	See section	60.7.113/ 3032
ACE	Message Buffer Index Register (FR_MBIDXR89)	16	R/W	See section	60.7.114/ 3032
AD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR90)	16	R/W	0000h	60.7.111/ 3028
AD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR90)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AD4	Message Buffer Frame ID Register (FR_MBFIDR90)	16	R/W	See section	60.7.113/ 3032
AD6	Message Buffer Index Register (FR_MBIDXR90)	16	R/W	See section	60.7.114/ 3032
AD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR91)	16	R/W	0000h	60.7.111/ 3028
ADA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR91)	16	R/W	See section	60.7.112/ 3030
ADC	Message Buffer Frame ID Register (FR_MBFIDR91)	16	R/W	See section	60.7.113/ 3032
ADE	Message Buffer Index Register (FR_MBIDXR91)	16	R/W	See section	60.7.114/ 3032
AE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR92)	16	R/W	0000h	60.7.111/ 3028
AE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR92)	16	R/W	See section	60.7.112/ 3030
AE4	Message Buffer Frame ID Register (FR_MBFIDR92)	16	R/W	See section	60.7.113/ 3032
AE6	Message Buffer Index Register (FR_MBIDXR92)	16	R/W	See section	60.7.114/ 3032
AE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR93)	16	R/W	0000h	60.7.111/ 3028
AEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR93)	16	R/W	See section	60.7.112/ 3030
AEC	Message Buffer Frame ID Register (FR_MBFIDR93)	16	R/W	See section	60.7.113/ 3032
AEE	Message Buffer Index Register (FR_MBIDXR93)	16	R/W	See section	60.7.114/ 3032
AF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR94)	16	R/W	0000h	60.7.111/ 3028
AF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR94)	16	R/W	See section	60.7.112/ 3030
AF4	Message Buffer Frame ID Register (FR_MBFIDR94)	16	R/W	See section	60.7.113/ 3032
AF6	Message Buffer Index Register (FR_MBIDXR94)	16	R/W	See section	60.7.114/ 3032
AF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR95)	16	R/W	0000h	60.7.111/ 3028
AFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR95)	16	R/W	See section	60.7.112/ 3030
AFC	Message Buffer Frame ID Register (FR_MBFIDR95)	16	R/W	See section	60.7.113/ 3032
AFE	Message Buffer Index Register (FR_MBIDXR95)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR96)	16	R/W	0000h	60.7.111/ 3028
B02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR96)	16	R/W	See section	60.7.112/ 3030
B04	Message Buffer Frame ID Register (FR_MBFIDR96)	16	R/W	See section	60.7.113/ 3032
B06	Message Buffer Index Register (FR_MBIDXR96)	16	R/W	See section	60.7.114/ 3032
B08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR97)	16	R/W	0000h	60.7.111/ 3028
B0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR97)	16	R/W	See section	60.7.112/ 3030
B0C	Message Buffer Frame ID Register (FR_MBFIDR97)	16	R/W	See section	60.7.113/ 3032
B0E	Message Buffer Index Register (FR_MBIDXR97)	16	R/W	See section	60.7.114/ 3032
B10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR98)	16	R/W	0000h	60.7.111/ 3028
B12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR98)	16	R/W	See section	60.7.112/ 3030
B14	Message Buffer Frame ID Register (FR_MBFIDR98)	16	R/W	See section	60.7.113/ 3032
B16	Message Buffer Index Register (FR_MBIDXR98)	16	R/W	See section	60.7.114/ 3032
B18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR99)	16	R/W	0000h	60.7.111/ 3028
B1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR99)	16	R/W	See section	60.7.112/ 3030
B1C	Message Buffer Frame ID Register (FR_MBFIDR99)	16	R/W	See section	60.7.113/ 3032
B1E	Message Buffer Index Register (FR_MBIDXR99)	16	R/W	See section	60.7.114/ 3032
B20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR100)	16	R/W	0000h	60.7.111/ 3028
B22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR100)	16	R/W	See section	60.7.112/ 3030
B24	Message Buffer Frame ID Register (FR_MBFIDR100)	16	R/W	See section	60.7.113/ 3032
B26	Message Buffer Index Register (FR_MBIDXR100)	16	R/W	See section	60.7.114/ 3032
B28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR101)	16	R/W	0000h	60.7.111/ 3028
B2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR101)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B2C	Message Buffer Frame ID Register (FR_MBFIDR101)	16	R/W	See section	60.7.113/ 3032
B2E	Message Buffer Index Register (FR_MBIDXR101)	16	R/W	See section	60.7.114/ 3032
B30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR102)	16	R/W	0000h	60.7.111/ 3028
B32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR102)	16	R/W	See section	60.7.112/ 3030
B34	Message Buffer Frame ID Register (FR_MBFIDR102)	16	R/W	See section	60.7.113/ 3032
B36	Message Buffer Index Register (FR_MBIDXR102)	16	R/W	See section	60.7.114/ 3032
B38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR103)	16	R/W	0000h	60.7.111/ 3028
B3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR103)	16	R/W	See section	60.7.112/ 3030
B3C	Message Buffer Frame ID Register (FR_MBFIDR103)	16	R/W	See section	60.7.113/ 3032
B3E	Message Buffer Index Register (FR_MBIDXR103)	16	R/W	See section	60.7.114/ 3032
B40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR104)	16	R/W	0000h	60.7.111/ 3028
B42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR104)	16	R/W	See section	60.7.112/ 3030
B44	Message Buffer Frame ID Register (FR_MBFIDR104)	16	R/W	See section	60.7.113/ 3032
B46	Message Buffer Index Register (FR_MBIDXR104)	16	R/W	See section	60.7.114/ 3032
B48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR105)	16	R/W	0000h	60.7.111/ 3028
B4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR105)	16	R/W	See section	60.7.112/ 3030
B4C	Message Buffer Frame ID Register (FR_MBFIDR105)	16	R/W	See section	60.7.113/ 3032
B4E	Message Buffer Index Register (FR_MBIDXR105)	16	R/W	See section	60.7.114/ 3032
B50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR106)	16	R/W	0000h	60.7.111/ 3028
B52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR106)	16	R/W	See section	60.7.112/ 3030
B54	Message Buffer Frame ID Register (FR_MBFIDR106)	16	R/W	See section	60.7.113/ 3032
B56	Message Buffer Index Register (FR_MBIDXR106)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR107)	16	R/W	0000h	60.7.111/ 3028
B5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR107)	16	R/W	See section	60.7.112/ 3030
B5C	Message Buffer Frame ID Register (FR_MBFIDR107)	16	R/W	See section	60.7.113/ 3032
B5E	Message Buffer Index Register (FR_MBIDXR107)	16	R/W	See section	60.7.114/ 3032
B60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR108)	16	R/W	0000h	60.7.111/ 3028
B62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR108)	16	R/W	See section	60.7.112/ 3030
B64	Message Buffer Frame ID Register (FR_MBFIDR108)	16	R/W	See section	60.7.113/ 3032
B66	Message Buffer Index Register (FR_MBIDXR108)	16	R/W	See section	60.7.114/ 3032
B68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR109)	16	R/W	0000h	60.7.111/ 3028
B6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR109)	16	R/W	See section	60.7.112/ 3030
B6C	Message Buffer Frame ID Register (FR_MBFIDR109)	16	R/W	See section	60.7.113/ 3032
B6E	Message Buffer Index Register (FR_MBIDXR109)	16	R/W	See section	60.7.114/ 3032
B70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR110)	16	R/W	0000h	60.7.111/ 3028
B72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR110)	16	R/W	See section	60.7.112/ 3030
B74	Message Buffer Frame ID Register (FR_MBFIDR110)	16	R/W	See section	60.7.113/ 3032
B76	Message Buffer Index Register (FR_MBIDXR110)	16	R/W	See section	60.7.114/ 3032
B78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR111)	16	R/W	0000h	60.7.111/ 3028
B7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR111)	16	R/W	See section	60.7.112/ 3030
B7C	Message Buffer Frame ID Register (FR_MBFIDR111)	16	R/W	See section	60.7.113/ 3032
B7E	Message Buffer Index Register (FR_MBIDXR111)	16	R/W	See section	60.7.114/ 3032
B80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR112)	16	R/W	0000h	60.7.111/ 3028
B82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR112)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B84	Message Buffer Frame ID Register (FR_MBFIDR112)	16	R/W	See section	60.7.113/ 3032
B86	Message Buffer Index Register (FR_MBIDXR112)	16	R/W	See section	60.7.114/ 3032
B88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR113)	16	R/W	0000h	60.7.111/ 3028
B8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR113)	16	R/W	See section	60.7.112/ 3030
B8C	Message Buffer Frame ID Register (FR_MBFIDR113)	16	R/W	See section	60.7.113/ 3032
B8E	Message Buffer Index Register (FR_MBIDXR113)	16	R/W	See section	60.7.114/ 3032
B90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR114)	16	R/W	0000h	60.7.111/ 3028
B92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR114)	16	R/W	See section	60.7.112/ 3030
B94	Message Buffer Frame ID Register (FR_MBFIDR114)	16	R/W	See section	60.7.113/ 3032
B96	Message Buffer Index Register (FR_MBIDXR114)	16	R/W	See section	60.7.114/ 3032
B98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR115)	16	R/W	0000h	60.7.111/ 3028
B9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR115)	16	R/W	See section	60.7.112/ 3030
B9C	Message Buffer Frame ID Register (FR_MBFIDR115)	16	R/W	See section	60.7.113/ 3032
B9E	Message Buffer Index Register (FR_MBIDXR115)	16	R/W	See section	60.7.114/ 3032
BA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR116)	16	R/W	0000h	60.7.111/ 3028
BA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR116)	16	R/W	See section	60.7.112/ 3030
BA4	Message Buffer Frame ID Register (FR_MBFIDR116)	16	R/W	See section	60.7.113/ 3032
BA6	Message Buffer Index Register (FR_MBIDXR116)	16	R/W	See section	60.7.114/ 3032
BA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR117)	16	R/W	0000h	60.7.111/ 3028
BAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR117)	16	R/W	See section	60.7.112/ 3030
BAC	Message Buffer Frame ID Register (FR_MBFIDR117)	16	R/W	See section	60.7.113/ 3032
BAE	Message Buffer Index Register (FR_MBIDXR117)	16	R/W	See section	60.7.114/ 3032

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR118)	16	R/W	0000h	60.7.111/ 3028
BB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR118)	16	R/W	See section	60.7.112/ 3030
BB4	Message Buffer Frame ID Register (FR_MBFIDR118)	16	R/W	See section	60.7.113/ 3032
BB6	Message Buffer Index Register (FR_MBIDXR118)	16	R/W	See section	60.7.114/ 3032
BB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR119)	16	R/W	0000h	60.7.111/ 3028
BBA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR119)	16	R/W	See section	60.7.112/ 3030
BBC	Message Buffer Frame ID Register (FR_MBFIDR119)	16	R/W	See section	60.7.113/ 3032
BBE	Message Buffer Index Register (FR_MBIDXR119)	16	R/W	See section	60.7.114/ 3032
BC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR120)	16	R/W	0000h	60.7.111/ 3028
BC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR120)	16	R/W	See section	60.7.112/ 3030
BC4	Message Buffer Frame ID Register (FR_MBFIDR120)	16	R/W	See section	60.7.113/ 3032
BC6	Message Buffer Index Register (FR_MBIDXR120)	16	R/W	See section	60.7.114/ 3032
BC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR121)	16	R/W	0000h	60.7.111/ 3028
BCA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR121)	16	R/W	See section	60.7.112/ 3030
BCC	Message Buffer Frame ID Register (FR_MBFIDR121)	16	R/W	See section	60.7.113/ 3032
BCE	Message Buffer Index Register (FR_MBIDXR121)	16	R/W	See section	60.7.114/ 3032
BD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR122)	16	R/W	0000h	60.7.111/ 3028
BD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR122)	16	R/W	See section	60.7.112/ 3030
BD4	Message Buffer Frame ID Register (FR_MBFIDR122)	16	R/W	See section	60.7.113/ 3032
BD6	Message Buffer Index Register (FR_MBIDXR122)	16	R/W	See section	60.7.114/ 3032
BD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR123)	16	R/W	0000h	60.7.111/ 3028
BDA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR123)	16	R/W	See section	60.7.112/ 3030

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BDC	Message Buffer Frame ID Register (FR_MBFIDR123)	16	R/W	See section	60.7.113/ 3032
BDE	Message Buffer Index Register (FR_MBIDXR123)	16	R/W	See section	60.7.114/ 3032
BE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR124)	16	R/W	0000h	60.7.111/ 3028
BE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR124)	16	R/W	See section	60.7.112/ 3030
BE4	Message Buffer Frame ID Register (FR_MBFIDR124)	16	R/W	See section	60.7.113/ 3032
BE6	Message Buffer Index Register (FR_MBIDXR124)	16	R/W	See section	60.7.114/ 3032
BE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR125)	16	R/W	0000h	60.7.111/ 3028
BEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR125)	16	R/W	See section	60.7.112/ 3030
BEC	Message Buffer Frame ID Register (FR_MBFIDR125)	16	R/W	See section	60.7.113/ 3032
BEE	Message Buffer Index Register (FR_MBIDXR125)	16	R/W	See section	60.7.114/ 3032
BF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR126)	16	R/W	0000h	60.7.111/ 3028
BF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR126)	16	R/W	See section	60.7.112/ 3030
BF4	Message Buffer Frame ID Register (FR_MBFIDR126)	16	R/W	See section	60.7.113/ 3032
BF6	Message Buffer Index Register (FR_MBIDXR126)	16	R/W	See section	60.7.114/ 3032
BF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR127)	16	R/W	0000h	60.7.111/ 3028
BFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR127)	16	R/W	See section	60.7.112/ 3030
BFC	Message Buffer Frame ID Register (FR_MBFIDR127)	16	R/W	See section	60.7.113/ 3032
BFE	Message Buffer Index Register (FR_MBIDXR127)	16	R/W	See section	60.7.114/ 3032
1000	Message Buffer Data Field Offset Register (FR_MBDOR0)	16	R/W	See section	60.7.115/ 3033
1002	Message Buffer Data Field Offset Register (FR_MBDOR1)	16	R/W	See section	60.7.115/ 3033
1004	Message Buffer Data Field Offset Register (FR_MBDOR2)	16	R/W	See section	60.7.115/ 3033
1006	Message Buffer Data Field Offset Register (FR_MBDOR3)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1008	Message Buffer Data Field Offset Register (FR_MBDOR4)	16	R/W	See section	60.7.115/ 3033
100A	Message Buffer Data Field Offset Register (FR_MBDOR5)	16	R/W	See section	60.7.115/ 3033
100C	Message Buffer Data Field Offset Register (FR_MBDOR6)	16	R/W	See section	60.7.115/ 3033
100E	Message Buffer Data Field Offset Register (FR_MBDOR7)	16	R/W	See section	60.7.115/ 3033
1010	Message Buffer Data Field Offset Register (FR_MBDOR8)	16	R/W	See section	60.7.115/ 3033
1012	Message Buffer Data Field Offset Register (FR_MBDOR9)	16	R/W	See section	60.7.115/ 3033
1014	Message Buffer Data Field Offset Register (FR_MBDOR10)	16	R/W	See section	60.7.115/ 3033
1016	Message Buffer Data Field Offset Register (FR_MBDOR11)	16	R/W	See section	60.7.115/ 3033
1018	Message Buffer Data Field Offset Register (FR_MBDOR12)	16	R/W	See section	60.7.115/ 3033
101A	Message Buffer Data Field Offset Register (FR_MBDOR13)	16	R/W	See section	60.7.115/ 3033
101C	Message Buffer Data Field Offset Register (FR_MBDOR14)	16	R/W	See section	60.7.115/ 3033
101E	Message Buffer Data Field Offset Register (FR_MBDOR15)	16	R/W	See section	60.7.115/ 3033
1020	Message Buffer Data Field Offset Register (FR_MBDOR16)	16	R/W	See section	60.7.115/ 3033
1022	Message Buffer Data Field Offset Register (FR_MBDOR17)	16	R/W	See section	60.7.115/ 3033
1024	Message Buffer Data Field Offset Register (FR_MBDOR18)	16	R/W	See section	60.7.115/ 3033
1026	Message Buffer Data Field Offset Register (FR_MBDOR19)	16	R/W	See section	60.7.115/ 3033
1028	Message Buffer Data Field Offset Register (FR_MBDOR20)	16	R/W	See section	60.7.115/ 3033
102A	Message Buffer Data Field Offset Register (FR_MBDOR21)	16	R/W	See section	60.7.115/ 3033
102C	Message Buffer Data Field Offset Register (FR_MBDOR22)	16	R/W	See section	60.7.115/ 3033
102E	Message Buffer Data Field Offset Register (FR_MBDOR23)	16	R/W	See section	60.7.115/ 3033
1030	Message Buffer Data Field Offset Register (FR_MBDOR24)	16	R/W	See section	60.7.115/ 3033
1032	Message Buffer Data Field Offset Register (FR_MBDOR25)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1034	Message Buffer Data Field Offset Register (FR_MBDOR26)	16	R/W	See section	60.7.115/ 3033
1036	Message Buffer Data Field Offset Register (FR_MBDOR27)	16	R/W	See section	60.7.115/ 3033
1038	Message Buffer Data Field Offset Register (FR_MBDOR28)	16	R/W	See section	60.7.115/ 3033
103A	Message Buffer Data Field Offset Register (FR_MBDOR29)	16	R/W	See section	60.7.115/ 3033
103C	Message Buffer Data Field Offset Register (FR_MBDOR30)	16	R/W	See section	60.7.115/ 3033
103E	Message Buffer Data Field Offset Register (FR_MBDOR31)	16	R/W	See section	60.7.115/ 3033
1040	Message Buffer Data Field Offset Register (FR_MBDOR32)	16	R/W	See section	60.7.115/ 3033
1042	Message Buffer Data Field Offset Register (FR_MBDOR33)	16	R/W	See section	60.7.115/ 3033
1044	Message Buffer Data Field Offset Register (FR_MBDOR34)	16	R/W	See section	60.7.115/ 3033
1046	Message Buffer Data Field Offset Register (FR_MBDOR35)	16	R/W	See section	60.7.115/ 3033
1048	Message Buffer Data Field Offset Register (FR_MBDOR36)	16	R/W	See section	60.7.115/ 3033
104A	Message Buffer Data Field Offset Register (FR_MBDOR37)	16	R/W	See section	60.7.115/ 3033
104C	Message Buffer Data Field Offset Register (FR_MBDOR38)	16	R/W	See section	60.7.115/ 3033
104E	Message Buffer Data Field Offset Register (FR_MBDOR39)	16	R/W	See section	60.7.115/ 3033
1050	Message Buffer Data Field Offset Register (FR_MBDOR40)	16	R/W	See section	60.7.115/ 3033
1052	Message Buffer Data Field Offset Register (FR_MBDOR41)	16	R/W	See section	60.7.115/ 3033
1054	Message Buffer Data Field Offset Register (FR_MBDOR42)	16	R/W	See section	60.7.115/ 3033
1056	Message Buffer Data Field Offset Register (FR_MBDOR43)	16	R/W	See section	60.7.115/ 3033
1058	Message Buffer Data Field Offset Register (FR_MBDOR44)	16	R/W	See section	60.7.115/ 3033
105A	Message Buffer Data Field Offset Register (FR_MBDOR45)	16	R/W	See section	60.7.115/ 3033
105C	Message Buffer Data Field Offset Register (FR_MBDOR46)	16	R/W	See section	60.7.115/ 3033
105E	Message Buffer Data Field Offset Register (FR_MBDOR47)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1060	Message Buffer Data Field Offset Register (FR_MBDOR48)	16	R/W	See section	60.7.115/ 3033
1062	Message Buffer Data Field Offset Register (FR_MBDOR49)	16	R/W	See section	60.7.115/ 3033
1064	Message Buffer Data Field Offset Register (FR_MBDOR50)	16	R/W	See section	60.7.115/ 3033
1066	Message Buffer Data Field Offset Register (FR_MBDOR51)	16	R/W	See section	60.7.115/ 3033
1068	Message Buffer Data Field Offset Register (FR_MBDOR52)	16	R/W	See section	60.7.115/ 3033
106A	Message Buffer Data Field Offset Register (FR_MBDOR53)	16	R/W	See section	60.7.115/ 3033
106C	Message Buffer Data Field Offset Register (FR_MBDOR54)	16	R/W	See section	60.7.115/ 3033
106E	Message Buffer Data Field Offset Register (FR_MBDOR55)	16	R/W	See section	60.7.115/ 3033
1070	Message Buffer Data Field Offset Register (FR_MBDOR56)	16	R/W	See section	60.7.115/ 3033
1072	Message Buffer Data Field Offset Register (FR_MBDOR57)	16	R/W	See section	60.7.115/ 3033
1074	Message Buffer Data Field Offset Register (FR_MBDOR58)	16	R/W	See section	60.7.115/ 3033
1076	Message Buffer Data Field Offset Register (FR_MBDOR59)	16	R/W	See section	60.7.115/ 3033
1078	Message Buffer Data Field Offset Register (FR_MBDOR60)	16	R/W	See section	60.7.115/ 3033
107A	Message Buffer Data Field Offset Register (FR_MBDOR61)	16	R/W	See section	60.7.115/ 3033
107C	Message Buffer Data Field Offset Register (FR_MBDOR62)	16	R/W	See section	60.7.115/ 3033
107E	Message Buffer Data Field Offset Register (FR_MBDOR63)	16	R/W	See section	60.7.115/ 3033
1080	Message Buffer Data Field Offset Register (FR_MBDOR64)	16	R/W	See section	60.7.115/ 3033
1082	Message Buffer Data Field Offset Register (FR_MBDOR65)	16	R/W	See section	60.7.115/ 3033
1084	Message Buffer Data Field Offset Register (FR_MBDOR66)	16	R/W	See section	60.7.115/ 3033
1086	Message Buffer Data Field Offset Register (FR_MBDOR67)	16	R/W	See section	60.7.115/ 3033
1088	Message Buffer Data Field Offset Register (FR_MBDOR68)	16	R/W	See section	60.7.115/ 3033
108A	Message Buffer Data Field Offset Register (FR_MBDOR69)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
108C	Message Buffer Data Field Offset Register (FR_MBDOR70)	16	R/W	See section	60.7.115/ 3033
108E	Message Buffer Data Field Offset Register (FR_MBDOR71)	16	R/W	See section	60.7.115/ 3033
1090	Message Buffer Data Field Offset Register (FR_MBDOR72)	16	R/W	See section	60.7.115/ 3033
1092	Message Buffer Data Field Offset Register (FR_MBDOR73)	16	R/W	See section	60.7.115/ 3033
1094	Message Buffer Data Field Offset Register (FR_MBDOR74)	16	R/W	See section	60.7.115/ 3033
1096	Message Buffer Data Field Offset Register (FR_MBDOR75)	16	R/W	See section	60.7.115/ 3033
1098	Message Buffer Data Field Offset Register (FR_MBDOR76)	16	R/W	See section	60.7.115/ 3033
109A	Message Buffer Data Field Offset Register (FR_MBDOR77)	16	R/W	See section	60.7.115/ 3033
109C	Message Buffer Data Field Offset Register (FR_MBDOR78)	16	R/W	See section	60.7.115/ 3033
109E	Message Buffer Data Field Offset Register (FR_MBDOR79)	16	R/W	See section	60.7.115/ 3033
10A0	Message Buffer Data Field Offset Register (FR_MBDOR80)	16	R/W	See section	60.7.115/ 3033
10A2	Message Buffer Data Field Offset Register (FR_MBDOR81)	16	R/W	See section	60.7.115/ 3033
10A4	Message Buffer Data Field Offset Register (FR_MBDOR82)	16	R/W	See section	60.7.115/ 3033
10A6	Message Buffer Data Field Offset Register (FR_MBDOR83)	16	R/W	See section	60.7.115/ 3033
10A8	Message Buffer Data Field Offset Register (FR_MBDOR84)	16	R/W	See section	60.7.115/ 3033
10AA	Message Buffer Data Field Offset Register (FR_MBDOR85)	16	R/W	See section	60.7.115/ 3033
10AC	Message Buffer Data Field Offset Register (FR_MBDOR86)	16	R/W	See section	60.7.115/ 3033
10AE	Message Buffer Data Field Offset Register (FR_MBDOR87)	16	R/W	See section	60.7.115/ 3033
10B0	Message Buffer Data Field Offset Register (FR_MBDOR88)	16	R/W	See section	60.7.115/ 3033
10B2	Message Buffer Data Field Offset Register (FR_MBDOR89)	16	R/W	See section	60.7.115/ 3033
10B4	Message Buffer Data Field Offset Register (FR_MBDOR90)	16	R/W	See section	60.7.115/ 3033
10B6	Message Buffer Data Field Offset Register (FR_MBDOR91)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10B8	Message Buffer Data Field Offset Register (FR_MBDOR92)	16	R/W	See section	60.7.115/ 3033
10BA	Message Buffer Data Field Offset Register (FR_MBDOR93)	16	R/W	See section	60.7.115/ 3033
10BC	Message Buffer Data Field Offset Register (FR_MBDOR94)	16	R/W	See section	60.7.115/ 3033
10BE	Message Buffer Data Field Offset Register (FR_MBDOR95)	16	R/W	See section	60.7.115/ 3033
10C0	Message Buffer Data Field Offset Register (FR_MBDOR96)	16	R/W	See section	60.7.115/ 3033
10C2	Message Buffer Data Field Offset Register (FR_MBDOR97)	16	R/W	See section	60.7.115/ 3033
10C4	Message Buffer Data Field Offset Register (FR_MBDOR98)	16	R/W	See section	60.7.115/ 3033
10C6	Message Buffer Data Field Offset Register (FR_MBDOR99)	16	R/W	See section	60.7.115/ 3033
10C8	Message Buffer Data Field Offset Register (FR_MBDOR100)	16	R/W	See section	60.7.115/ 3033
10CA	Message Buffer Data Field Offset Register (FR_MBDOR101)	16	R/W	See section	60.7.115/ 3033
10CC	Message Buffer Data Field Offset Register (FR_MBDOR102)	16	R/W	See section	60.7.115/ 3033
10CE	Message Buffer Data Field Offset Register (FR_MBDOR103)	16	R/W	See section	60.7.115/ 3033
10D0	Message Buffer Data Field Offset Register (FR_MBDOR104)	16	R/W	See section	60.7.115/ 3033
10D2	Message Buffer Data Field Offset Register (FR_MBDOR105)	16	R/W	See section	60.7.115/ 3033
10D4	Message Buffer Data Field Offset Register (FR_MBDOR106)	16	R/W	See section	60.7.115/ 3033
10D6	Message Buffer Data Field Offset Register (FR_MBDOR107)	16	R/W	See section	60.7.115/ 3033
10D8	Message Buffer Data Field Offset Register (FR_MBDOR108)	16	R/W	See section	60.7.115/ 3033
10DA	Message Buffer Data Field Offset Register (FR_MBDOR109)	16	R/W	See section	60.7.115/ 3033
10DC	Message Buffer Data Field Offset Register (FR_MBDOR110)	16	R/W	See section	60.7.115/ 3033
10DE	Message Buffer Data Field Offset Register (FR_MBDOR111)	16	R/W	See section	60.7.115/ 3033
10E0	Message Buffer Data Field Offset Register (FR_MBDOR112)	16	R/W	See section	60.7.115/ 3033
10E2	Message Buffer Data Field Offset Register (FR_MBDOR113)	16	R/W	See section	60.7.115/ 3033

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10E4	Message Buffer Data Field Offset Register (FR_MBDOR114)	16	R/W	See section	60.7.115/ 3033
10E6	Message Buffer Data Field Offset Register (FR_MBDOR115)	16	R/W	See section	60.7.115/ 3033
10E8	Message Buffer Data Field Offset Register (FR_MBDOR116)	16	R/W	See section	60.7.115/ 3033
10EA	Message Buffer Data Field Offset Register (FR_MBDOR117)	16	R/W	See section	60.7.115/ 3033
10EC	Message Buffer Data Field Offset Register (FR_MBDOR118)	16	R/W	See section	60.7.115/ 3033
10EE	Message Buffer Data Field Offset Register (FR_MBDOR119)	16	R/W	See section	60.7.115/ 3033
10F0	Message Buffer Data Field Offset Register (FR_MBDOR120)	16	R/W	See section	60.7.115/ 3033
10F2	Message Buffer Data Field Offset Register (FR_MBDOR121)	16	R/W	See section	60.7.115/ 3033
10F4	Message Buffer Data Field Offset Register (FR_MBDOR122)	16	R/W	See section	60.7.115/ 3033
10F6	Message Buffer Data Field Offset Register (FR_MBDOR123)	16	R/W	See section	60.7.115/ 3033
10F8	Message Buffer Data Field Offset Register (FR_MBDOR124)	16	R/W	See section	60.7.115/ 3033
10FA	Message Buffer Data Field Offset Register (FR_MBDOR125)	16	R/W	See section	60.7.115/ 3033
10FC	Message Buffer Data Field Offset Register (FR_MBDOR126)	16	R/W	See section	60.7.115/ 3033
10FE	Message Buffer Data Field Offset Register (FR_MBDOR127)	16	R/W	See section	60.7.115/ 3033
1100	Message Buffer Data Field Offset Register (FR_MBDOR128)	16	R/W	See section	60.7.115/ 3033
1102	Message Buffer Data Field Offset Register (FR_MBDOR129)	16	R/W	See section	60.7.115/ 3033
1104	Message Buffer Data Field Offset Register (FR_MBDOR130)	16	R/W	See section	60.7.115/ 3033
1106	Message Buffer Data Field Offset Register (FR_MBDOR131)	16	R/W	See section	60.7.115/ 3033
1108	LRAM ECC Error Test Register (FR_LEETR0)	16	R/W	See section	60.7.116/ 3034
110A	LRAM ECC Error Test Register (FR_LEETR1)	16	R/W	See section	60.7.116/ 3034
110C	LRAM ECC Error Test Register (FR_LEETR2)	16	R/W	See section	60.7.116/ 3034
110E	LRAM ECC Error Test Register (FR_LEETR3)	16	R/W	See section	60.7.116/ 3034

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1110	LRAM ECC Error Test Register (FR_LEETR4)	16	R/W	See section	60.7.116/ 3034
1112	LRAM ECC Error Test Register (FR_LEETR5)	16	R/W	See section	60.7.116/ 3034

60.7.1 Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHIVER								PEVER							
Write																
Reset	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0

FR_MVR field descriptions

Field	Description
0–7 CHIVER	CHI Version Number. This field provides the version number of the controller host interface.
8–15 PEVER	PE Version Number. This field provides the version number of the protocol engine.

60.7.2 Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Write: SBFF, SCM, CHB, CHA, ECCE, FUM, FAM , CLKSEL , BITRATE: Disabled Mode and SFFE, MEN: Disabled Mode or POC: config

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read Write	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	Reserved
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FUM	FAM	0	CLKSEL	BITRATE			0
Write								
Reset	0	0	0	0	0	0	0	0

FR_MCR field descriptions

Field	Description
0 MEN	Module Enable. This bit indicates whether or not the CC is in the Disabled Mode. The application requests the CC to leave the Disabled Mode by writing 1 to this bit Before leaving the Disabled Mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, BITRATE values. For details see Modes of Operation. **NOTE:** If the CC is enabled it can only be disabled during mode: POC:default config. 0 Write: only during POC:default config, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled
1 SBFF	System Bus Failure Freeze This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to freeze mode
2 SCM	Single Channel Device Mode. This control bit defines the channel device mode of the CC as described in Channel Device Modes. 0 CC works in dual channel device mode 1 CC works in single channel device mode
3 CHB	Channel B Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the following table.

Table 60-8. FlexRay channel selection

SCM	CHB	CHA	Description
Dual Channel Device Modes			
0	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	1	0	reserved
	1	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
Single Channel Device Mode			
1	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A

Table continues on the next page...

FR_MCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 60-8. FlexRay channel selection (continued)
<table>
<tr><th>SCM</th><th>CHB</th><th>CHA</th><th>Description</th></tr>
<tr><td rowspan="3"></td><td></td><td></td><td>ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC</td></tr>
<tr><td>1</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table>
</td></tr>
<tr><td>4
CHA</td><td>Channel A Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the FlexRay channel selection table as seen above.</td></tr>
<tr><td>5
SFFE</td><td>Synchronization Frame Filter Enable. This bit controls the filtering for received synchronization frames. For details see Sync Frame Filtering .

0 Synchronization frame filtering disabled
1 Synchronization frame filtering enabled</td></tr>
<tr><td>6
ECCE</td><td>ECC Functionality Enable. This bit controls the ECC memory error detection functionality. For details see Memory Content Fault Detection .

0 ECC functionality (injection, detection, reporting, response) disabled
1 ECC functionality enabled</td></tr>
<tr><td>7
Reserved</td><td>Reserved bit, will not be changed. Application must not write any value different from the reset value.

This field is reserved.</td></tr>
<tr><td>8
FUM</td><td>FIFO Update Mode. This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see FIFO Update)

0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF]
1 FIFOA/FIFOB) is not updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]</td></tr>
<tr><td>9
FAM</td><td>FIFO Address Mode. This bit controls the location of the system memory base address for the FIFOs. (see FIFO Configuration)

0 FIFO Base Address located in System Memory Base Address High Register (FR_SYMBADHR))
1 FIFO Base Address located in Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)</td></tr>
<tr><td>10
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11
CLKSEL</td><td>Protocol Engine Clock Source Select. This bit is used to select the clock source for the protocol engine.

0 PE clock source is generated by on-chip crystal oscillator.
1 PE clock source is generated by on-chip PLL.</td></tr>
<tr><td>12–14
BITRATE</td><td>FlexRay Bus Bit Rate. This bit field defines the FlexRay Bus Bit Rate.

000 10.0 Mbit/sec
001 5.0 Mbit/sec
010 2.5 Mbit/sec
011 8.0 Mbit/sec</td></tr>
</table>

Table continues on the next page...

FR_MCR field descriptions (continued)

Field	Description
	100 reserved 101 reserved 110 reserved 111 reserved
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.3 System Memory Base Address High Register (FR_SYMBADHR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADHR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADHR register contains the most significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address high. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

60.7.4 System Memory Base Address Low Register (FR_SYMBADLR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADLR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADLR register contains the least significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SMBA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address low. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.5 Strobe Signal Control Register (FR_STBSCR)

Write: Anytime

This register is used to assign the individual protocol timing related strobe signals given in Table 60-9 to the external strobe ports. Each strobe signal can be assigned to at most one strobe port . Each write access to registers overwrites the previously written ENB and STBPSEL values for the signal indicated by SEL . If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port . If no strobe signal is assigned to a strobe port, the strobe port carries logic 0 . For more detailed and timing information refer to Strobe Signal Support .

NOTE

In single channel device mode, channel B related strobe signals are undefined and should not be assigned to the strobe ports.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			SEL				0			ENB	0		STBPSEL	
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STBSCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SEL	Strobe Signal Select. This control field selects one of the strobe signals given in Table 60-9 to be enabled or disabled and assigned to one of the four strobe ports given in the following table.

Table 60-9. Strobe Signal Mapping

SEL		Description	Channel	Type	Offset (Given in PE clock cycles)	Reference
dec	hex					
0	0x0	arm	-	value	+1	MT start
1	0x1	mt	-	value	+1	MT start
2	0x2	cycle start	-	pulse	0	MT start
3	0x3	minislot start	-	pulse	0	MT start
4	0x4	slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX
12	0xC	content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX
14	0xE	receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt
15	0xF		B			RX FIFO B Almost Full Interrupt

Write: Disabled Mode

Table continues on the next page...

FR_STBSCR field descriptions (continued)

Field	Description
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ENB	Strobe Signal Enable. The control bit is used to enable and to disable the strobe signal selected by STBSSEL. 0 Strobe signal is disabled and not assigned to any strobe port. 1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 STBPSEL	Strobe Port Select. This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation. 00 assign selected signal to FR_DBG[0] 01 assign selected signal to FR_DBG[1] 10 assign selected signal to FR_DBG[2] 11 assign selected signal to FR_DBG[3]

60.7.6 Message Buffer Data Size Register (FR_MBDSR)

Write: POC:config

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	MBSEG2DS							0	MBSEG1DS						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBDSR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 MBSEG2DS	Message Buffer Segment 2 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the second message buffer segment.

Table continues on the next page...

FR_MBDSR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBSEG1DS	Message Buffer Segment 1 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the first message buffer segment.

60.7.7 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Write: POC:config

This register is used to define the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Address: 0h base + Eh offset = Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	LAST_MB_SEG1							0	LAST_MB_UTIL						
Write																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

FR_MBSSUTR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 LAST_MB_SEG1	Last Message Buffer In Segment 1. This field defines the message buffer number of the last individual message buffer that is assigned to the first message buffer segment. The individual message buffers in the first segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with n <= LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1+1 individual message buffers. **NOTE:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the second message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 128 . **NOTE:** If LAST_MB_SEG1 = 127 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LAST_MB_UTIL	Last Message Buffer Utilized. This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n <= LAST_MB_UTIL. **NOTE:** If LAST_MB_UTIL=LAST_MB_SEG1 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

60.7.8 PE DRAM Access Register (FR_PEDRAR)

The PEDRAR register is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this registers initiates a read or write operation on the PE DRAM. The access done status bit DAD is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR are written into the PE DRAM, read back from the PE DRAM and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data are read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description refer to Memory Content Fault Detection

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	INST				ADDR											DAD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRAR field descriptions

Field	Description
0–3 INST	PE DRAM Access Instruction. This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16 bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16 bit) into FR_PEDRDR[DATA] other reserved
4–14 ADDR	PE DRAM Access Address. This field defines the address in the PE DRAM to be written to or read from.
15 DAD	PE DRAM Access Done. This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done

60.7.9 PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Address: 0h base + 12h offset = 12h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRDR field descriptions

Field	Description
0–15 DATA	Data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

60.7.10 Protocol Operation Control Register (FR_POCR)

The application uses this register to issue

- protocol control commands
- external clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see Protocol Control Command Execution .

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to External Clock Synchronization .

Write: Normal Mode

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			EOC_AP		ERC_AP		BSY_ WMC	0			POCCMD			
Write	WME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_POCR field descriptions

Field	Description
0 WME	Write Mode External Correction. This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write. 1 No write to EOC_AP and ERC_AP fields on register write.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 EOC_AP	External Offset Correction Application. This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 do not apply external offset correction value 01 reserved 10 subtract external offset correction value 11 add external offset correction value
6–7 ERC_AP	External Rate Correction Application. This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21) 00 do not apply external rate correction value 01 reserved 10 subtract external rate correction value 11 add external rate correction value
8 BSY_WMC	**NOTE:** The name and function of this field varies depending on whether it is being read or written. BSY (Read-Only). Protocol Control Command Busy. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. WMC (Write-Only) Protocol Control Command Write. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. 0 (Write-Only) Write to POCCMD field on register write. 1 (Write-Only) Do not write to POCCMD field on register write. 0 (Read-Only) Command write idle, command accepted and ready to receive new protocol command. 1 (Read-Only) Command write busy, command not yet accepted, not ready to receive new protocol command.
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 POCCMD	Protocol Control Command. The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTART. Immediately activate capability of node to cold start cluster.

Table continues on the next page...

FR_POCR field descriptions (continued)

Field	Description
	0001 ALL_SLOTS. Delayed transition to the all slots transmission mode. (Delayed means on completion of current communication cycle.) 0010 CONFIG. Immediately transition to the POC:config state. 0011 FREEZE. Immediately transition to the POC:halt state. 0100 READY, CONFIG_COMPLETE. Immediately transition to the POC:ready state. 0101 RUN. Immediately transition to the POC:startup start state. 0110 DEFAULT_CONFIG. Immediately transition to the POC:default config state. 0111 HALT. Delayed transition to the POC:halt state 1000 WAKEUP. Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

60.7.11 Global Interrupt Flag and Enable Register (FR_GIFER)

Write: Normal Mode

This register provides the means to control some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables . The generation scheme for these flags is depicted in Figure 60-33 . For more details on interrupt generation, see Interrupt Support . These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write				w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
Reset	0	0	0	0	0	0	0	0

FR_GIFER field descriptions

Field	Description
0 MIF	Module Interrupt Flag. This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set.
1 PRIF	Protocol Interrupt Flag. This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set. 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set.
2 CHIF	CHI Interrupt Flag. This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the chi error interrupt enable bit FR_GIFER[CHIE] are set . 0 All CHI error flags are equal to 0 or the chi error interrupt is disabled. 1 At least one CHI error flag and the chi error interrupt enable are is set.
3 WUPIF	Wakeup Interrupt Flag. This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3). 0 No Wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus
4 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO B almost full event has occurred
5 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO A almost full event has occurred
6 RBIF	Receive Message Buffer Interrupt Flag. This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly, instead it is cleared by the CC when all of the interrupt flags MBIF of the individual receive message buffers are cleared by the application or if the application has cleared the related interrupt enables bit MBIE.

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
	0 None of the individual receive message buffers has the MBIF and MBIE flag set. 1 At least one individual receive message buffer has the MBIF and MBIE flag set.
7 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application can not clear this interrupt flag directly, instead, this interrupt flag is cleared by the CC when either all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application or the application has cleared the related interrupt enables bit MBIE. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set. 1 At least one individual transmit message buffer has the MBIF and MBIE flag set.
8 MIE	Module Interrupt Enable. This bit controls if the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	Protocol Interrupt Enable. This bit controls if the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
10 CHIE	CHI Interrupt Enable. This bit controls if the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	Wakeup Interrupt Enable. This bit controls if the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
12 FAFBIE	Receive FIFO Channel B Almost Full Interrupt Enable. This bit controls if the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	Receive FIFO Channel A Almost Full Interrupt Enable. This bit controls if the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	Receive Message Buffer Interrupt Enable. This bit controls if the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 TBIE	Transmit Message Buffer Interrupt Enable. This bit controls if the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

60.7.12 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	Fatal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. The fatal protocol errors are: 1) pLatestTx violation, as described in the MAC process of the FlexRay protocol 2) transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event. 1 Fatal protocol error detected.
1 INTL_IF	Internal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event. 1 Internal protocol error detected.
2 ILCF_IF	Illegal Protocol Configuration Interrupt Flag. This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the POC:halt state immediately. The protocol engine checks the listen_timeout value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of listen_timeout is equal to zero, the protocol configuration setting is considered as illegal. 0 No such event. 1 Illegal protocol configuration detected.
3 CSA_IF	Cold Start Abort Interrupt Flag. This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3).

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
	0 No such event. 1 Cold start aborted and no more coldstart attempts allowed.
4 MRC_IF	Missing Rate Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	Missing Offset Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event. 1 Insufficient number of measurements for offset correction detected.
6 CCL_IF	Clock Correction Limit Reached Interrupt Flag. This flag is set when the internal calculated offset or rate calculation values have reached or exceeded its configured thresholds as given by the offset_coorection_out field in the Protocol Configuration Register 9 (FR_PCR9) and the rate_correction_out field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event. 1 Offset or rate correction limit reached.
7 MXS_IF	Max Sync Frames Detected Interrupt Flag. This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the node_sync_max field in the Protocol Configuration Register 30 (FR_PCR30). **NOTE:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account. 0 No such event. 1 More than node_sync_max sync frames detected.
8 MTX_IF	Media Access Test Symbol Received Interrupt Flag. This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event. 1 MTS symbol received.
9 LTXB_IF	pLatestTx Violation on Channel B Interrupt Flag. This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation, as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel B.
10 LTXA_IF	pLatestTx Violation on Channel A Interrupt Flag. This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel A.
11 TBVB_IF	Transmission across boundary on channel B Interrupt Flag. This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol.

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
	0 No such event. 1 Transmission across boundary violation occurred on channel B.
12 TBVA_IF	Transmission across boundary on channel A Interrupt Flag. This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel A.
13 TI2_IF	Timer 2 Expired Interrupt Flag. This flag is set whenever timer 2 expires. 0 No such event. 1 Timer 2 has reached its time limit.
14 TI1_IF	Timer 1 Expired Interrupt Flag. This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	Cycle Start Interrupt Flag. This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started.

60.7.13 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IF	ODT_IF	0			
Write			w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	Error Mode Changed Interrupt Flag. This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC.

Table continues on the next page...

FR_PIFR1 field descriptions (continued)

Field	Description
	0 No such event. 1 ERRMODE field changed.
1 IPC_IF	Illegal Protocol Control Command Interrupt Flag. This flag is set when the PE tries to execute a protocol control command, which was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case the command is not executed. For more details, see Protocol Control Command Execution . 0 No such event. 1 Illegal protocol control command detected.
2 PECF_IF	Protocol Engine Communication Failure Interrupt Flag. This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event. 1 Protocol Engine Communication Failure detected.
3 PSC_IF	Protocol State Changed Interrupt Flag. This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event. 1 Protocol state changed.
4 SSI3_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
5 SSI2_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
6 SSI1_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
7 SSI0_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IF	Even Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the even cycle. 0 No such event. 1 Sync frame measurement table written
11 ODT_IF	Odd Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the odd cycle.

Table continues on the next page...

FR_PIFR1 field descriptions (continued)

Field	Description
	0 No such event. 1 Sync frame measurement table written
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.14 Protocol Interrupt Enable Register 0 (FR_PIER0)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read Write	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0

FR_PIER0 field descriptions

Field	Description
0 FATL_IE	Fatal Protocol Error Interrupt Enable. This bit controls FATL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 INTL_IE	Internal Protocol Error Interrupt Enable. This bit controls INTL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 ILCF_IE	Illegal Protocol Configuration Interrupt Enable. This bit controls ILCF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 CSA_IE	Cold Start Abort Interrupt Enable. This bit controls CSA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 MRC_IE	Missing Rate Correction Interrupt Enable. This bit controls MRC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
5 MOC_IE	Missing Offset Correction Interrupt Enable. This bit controls MOC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 CCL_IE	Clock Correction Limit Reached Interrupt Enable. This bit controls CCL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 MXS_IE	Max Sync Frames Detected Interrupt Enable. This bit controls MXS_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8 MTX_IE	Media Access Test Symbol Received Interrupt Enable. This bit controls MTX_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
9 LTXB_IE	pLatestTx Violation on Channel B Interrupt Enable. This bit controls LTXB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 LTXA_IE	pLatestTx Violation on Channel A Interrupt Enable. This bit controls LTXA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 TBVB_IE	Transmission across boundary on channel B Interrupt Enable. This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12 TBVA_IE	Transmission across boundary on channel A Interrupt Enable. This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
13 TI2_IE	Timer 2 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
14 TI1_IE	Timer 1 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
15 CYS_IE	Cycle Start Interrupt Enable. This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

60.7.15 Protocol Interrupt Enable Register 1 (FR_PIER1)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read Write	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IE	ODT_IE	0			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PIER1 field descriptions

Field	Description
0 EMC_IE	Error Mode Changed Interrupt Enable. This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 IPC_IE	Illegal Protocol Control Command Interrupt Enable. This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 PECF_IE	Protocol Engine Communication Failure Interrupt Enable. This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 PSC_IE	Protocol State Changed Interrupt Enable. This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 SSI3_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 SSI2_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 SSI1_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation.

Table continues on the next page...

FR_PIER1 field descriptions (continued)

Field	Description
	0 interrupt request generation disabled 1 interrupt request generation enabled
7 SSI0_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IE	Even Cycle Table Written Interrupt Enable. This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 ODT_IE	Odd Cycle Table Written Interrupt Enable. This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.16 CHI Error Flag Register (FR_CHIERFR)

Write: Normal Mode

This register holds the CHI related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit CHIE in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7
Read	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
Write		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	Frame Lost Channel B Error Flag. This flag is set if a complete frame was received on channel B but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost.

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
	0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	Frame Lost Channel A Error Flag. This flag is set if a complete frame was received on channel A but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected
2 PCMI_EF	Protocol Command Ignored Error Flag. This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	Receive FIFO Overrun Channel B Error Flag. This flag is set when an overrun of the FIFO for channel B occurred. This error occurs if a semantically valid frame was received on channel B and matches the all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
4 FOVA_EF	Receive FIFO Overrun Channel A Error Flag. This flag is set when an overrun of the FIFO for channel A occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	Message Buffer Search Error Flag. This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 131 is detected in the FR_MBIDXR[MBIDX] field of an found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to Message Buffer Search Error for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
6 MBU_EF	Message Buffer Utilization Error Flag. This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	Lock Error Flag. This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again.

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
	0 No such error 1 Lock error detected
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SBCF_EF	System Bus Communication Failure Error Flag. This flag is set if a system bus access was not finished within the required amount of time (see System Bus Access Timeout). 0 No such event 1 System bus access not finished in time
10 FID_EF	Frame ID Error Flag. This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register and "also an illegal CHI command RUN is issued in POC:normal active state". 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	Dynamic Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding protocol configuration register field max_payload_length_dynamic in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred
12 SPL_EF	Static Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding protocol configuration register field payload_length_static in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	Network Management Length Error Flag. This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
14 NMF_EF	Network Management Frame Error Flag. This flag is set if a received message in the static segment with a Preamble Indicator flag PP asserted has its Null Frame indicator flag NF asserted as well. In this case, the Global Network Management Registers (see Network Management Vector Registers (FR_NMVR0-FR_NMVR5)) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	Illegal System Bus Address Error Flag. This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see System Bus Illegal Address Access). 0 No such event 1 Illegal system bus address accessed

60.7.17 Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their interrupt status flag MBIF and interrupt enable MBIE bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Address: 0h base + 22h offset = 22h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	TBIVEC							0	RBIVEC						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBIVEC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 TBIVEC	Transmit Buffer Interrupt Vector. This field provides the number of the lowest numbered enabled transmit message buffer that has its interrupt status flag MBIF and its interrupt enable bit MBIE set. If there is no transmit message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 RBIVEC	Receive Buffer Interrupt Vector. This field provides the message buffer number of the lowest numbered receive message buffer which has its interrupt flag MBIF and its interrupt enable bit MBIE asserted. If there is no receive message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.

60.7.18 Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see Slot Status Monitoring.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHAERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CASERCR field descriptions

Field	Description
0–15 CHAERSCNT	Channel A Status Error Counter. This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

60.7.19 Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring see Slot Status Monitoring.

Address: 0h base + 26h offset = 26h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHBERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CBSERCR field descriptions

Field	Description
0–15 CHBERSCNT	Channel B Status Error Counter. This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

60.7.20 Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PSR0 field descriptions

Field	Description
0–1 ERRMODE	Error Mode. protocol related variable: vPOC!ErrorMode. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 reserved
2–3 SLOTMODE	Slot Mode. protocol related variable: vPOC!SlotMode. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 PROTSTATE	Protocol State. protocol related variable: vPOC!State. This field indicates the state of the protocol. 000 POC:default config 001 POC:config 010 POC:wakeup 011 POC:ready 100 POC:normal passive 101 POC:normal active 110 POC:halt 111 POC:startup

Table continues on the next page...

FR_PSR0 field descriptions (continued)

Field	Description
8–11 STARTUPSTATE	Startup State. protocol related variable: vPOC!StartupState. This field indicates the current sub-state of the startup procedure. 0000 reserved 0001 reserved 0010 POC:coldstart collision resolution 0011 POC:coldstart listen 0100 POC:integration consistency check 0101 POC:integration listen 0110 reserved 0111 POC:initialize schedule 1000 reserved 1001 reserved 1010 POC:coldstart consistency check 1011 reserved 1100 reserved 1101 POC:integration coldstart check 1110 POC:coldstart gap 1111 POC:coldstart join
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WAKEUPSTATUS	Wakeup Status. protocol related variable: vPOC!WakeupStatus. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_ HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 reserved

60.7.21 Protocol Status Register 1 (FR_PSR1)

Write: Normal Mode

Address: 0h base + 2Ah offset = 2Ah

Bit	0	1	2	3	4	5	6	7
Read	CSAA	CSP	0	REMCSAT				
Write	w1c							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CPN	HHR	FRZ	APTAC				
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR1 field descriptions

Field	Description
0 CSAA	Cold Start Attempt Aborted Flag. protocol related event: 'set coldstart abort indicator in CHI' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	Leading Cold Start Path. This status bit is set when the CC has reached the POC:normal active state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 POC:normal active reached from POC:startup state via leading cold start path
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 REMCSAT	Remaining Coldstart Attempts. protocol related variable: vRemainingColdstartAttempts This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	Leading Cold Start Path Noise. protocol related variable: vPOC!ColdstartNoise This status bit is set if the CC has reached the POC:normal active state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 POC:normal active state was reached from POC:startup state via noisy leading cold start path
9 HHR	Host Halt Request Pending. protocol related variable: vPOC!CHIHaltRequest This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register (FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 HALT command received

Table continues on the next page...

FR_PSR1 field descriptions (continued)

Field	Description
10 FRZ	Freeze Occurred. protocol related variable: vPOC!Freeze This status bit is set when the CC has reached the POC:halt state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11–15 APTAC	Allow Passive to Active Counter. protocol related variable: vPOC!vAllowPassivetoActive This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the POC:normal passive state due to an application configured delay to enter POC:normal active state. This delay is defined by the allow_passive_to_active field in the Protocol Configuration Register 12 (FR_PCR12).

60.7.22 Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the Network Idle Time NIT, the Symbol Window and the clock synchronization. The NIT related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The Symbol Window related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol window related status bits remain in their reset state. The clock synchronization related CLKCORRFAILCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7
Read	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	STCA	SBVA	SSEA	MTA	CKCORFCNT			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR2 field descriptions

Field	Description
0 NBVB	NIT Boundary Violation on Channel B. protocol related variable: vSS!BViolation for NIT on channel B This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	NIT Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for NIT on channel B This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	Symbol Window Transmit Conflict on Channel B. protocol related variable: vSS!TxConflict for symbol window on channel B This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	Symbol Window Boundary Violation on Channel B. protocol related variable: vSS!BViolation for symbol window on channel B This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
4 SSEB	Symbol Window Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for symbol window on channel B This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	Media Access Test Symbol MTS Received on Channel B. protocol related variable: vSS!ValidMTS for Symbol Window on channel B This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	NIT Boundary Violation on Channel A. protocol related variable: vSS!BViolation for NIT on channel A This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	NIT Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected

Table continues on the next page...

FR_PSR2 field descriptions (continued)

Field	Description
8 STCA	Symbol Window Transmit Conflict on Channel A. protocol related variable: vSS!TxConflict for symbol window on channel A This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	Symbol Window Boundary Violation on Channel A. protocol related variable: vSS!BViolation for symbol window on channel A This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	Symbol Window Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for symbol window on channel A This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected
11 MTA	Media Access Test Symbol MTS Received on Channel A. protocol related variable: vSS!ValidMTS for symbol window on channel A This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel A. 1 MTS symbol received 0 No such event
12–15 CKCORFCNT	Clock Correction Failed Counter. protocol related variable: vClockCorrectionFailed This field provides the number of consecutive even/odd communication cycle pairs that have passed without clock synchronization having performed an offset or a rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in the Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the POC:normal active state, or when both the rate and offset correction terms have been calculated successfully.

60.7.23 Protocol Status Register 3 (FR_PSR3)

Write: Normal Mode

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Address: 0h base + 2Eh offset = 2Eh

Bit	0	1	2	3	4	5	6	7
Read	0		WUB	ABVB	AACB	ACEB	ASEB	AVFB
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		WUA	ABVA	AACA	ACEA	ASEA	AVFA
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PSR3 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 WUB	Wakeup Symbol Received on Channel B. This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	Aggregated Boundary Violation on Channel B. This flag is set when a boundary violation has been detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
4 AACB	Aggregated Additional Communication on Channel B. This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	Aggregated Content Error on Channel B. This flag is set when a content error has been detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
6 ASEB	Aggregated Syntax Error on Channel B. This flag is set when a syntax error has been detected on channel B. Syntax errors are detected in the communication slots, the symbol window and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	Aggregated Valid Frame on Channel B. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 1 At least one syntactically valid frame received 0 No syntactically valid frames received
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_PSR3 field descriptions (continued)

Field	Description
10 WUA	Wakeup Symbol Received on Channel A. This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	Aggregated Boundary Violation on Channel A. This flag is set when a boundary violation has been detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	Aggregated Additional Communication on Channel A. This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	Aggregated Content Error on Channel A. This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
14 ASEA	Aggregated Syntax Error on Channel A. This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	Aggregated Valid Frame on Channel A. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

60.7.24 Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		MTCT													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

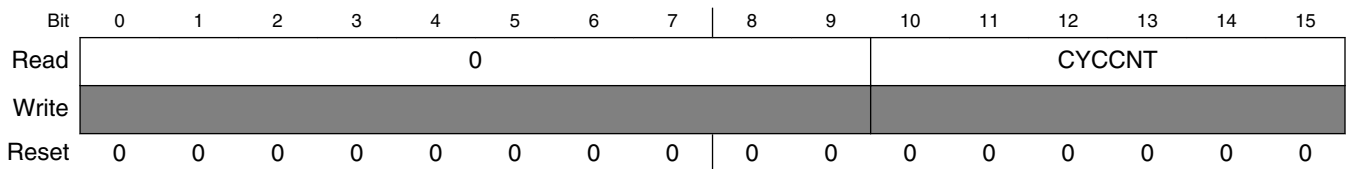
FR_MTCTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 MTCT	Macrotick Counter. protocol related variable: vMacrotick This field provides the macrotick count of the current communication cycle.

60.7.25 Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Address: 0h base + 32h offset = 32h



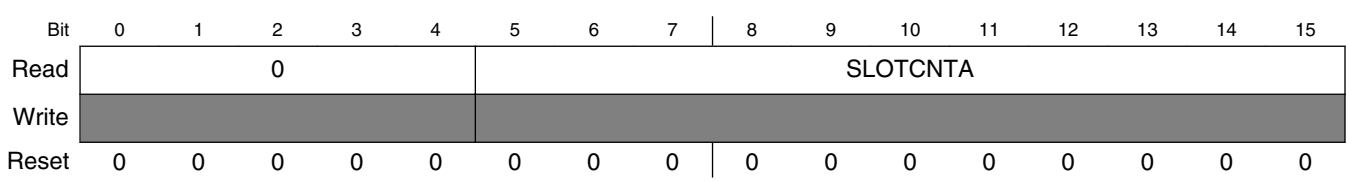
FR_CYCTR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNT	Cycle Counter. protocol related variable: vCycleCounter This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from zero again.

60.7.26 Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Address: 0h base + 34h offset = 34h



FR_SLTCTAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTA	Slot Counter Value for Channel A. protocol related variable: vSlotCounter for channel A This field provides the number of the current slot in the current communication cycle.

60.7.27 Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Address: 0h base + 36h offset = 36h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTB	Slot Counter Value for Channel B. protocol related variable: vSlotCounter for channel B This field provides the number of the current slot in the current communication cycle.

60.7.28 Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RATECORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RTCORVR field descriptions

Field	Description
0–15 RATECORR	Rate Correction Value. protocol related variable: vRateCorrection (before value limitation and external rate correction) This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

60.7.29 Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Address: 0h base + 3Ah offset = 3Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	OFFSETCORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_OFCORVR field descriptions

Field	Description
0–15 OFFSETCORR	Offset Correction Value. protocol related variable: vOffsetCorrection (before value limitation and external offset correction) This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

60.7.30 Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in Figure 60-35 . The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

NOTE

The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write								
Reset	0	0	0	0	0	0	0	0

FR_CIFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MIF	Module Interrupt Flag. This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	Protocol Interrupt Flag. This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	CHI Interrupt Flag. This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1.

Table continues on the next page...

FR_CIFR field descriptions (continued)

Field	Description
	0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	Wakeup Interrupt Flag. Provides the same value as FR_GIFER[WUPIF]
12 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFBIF]
13 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFAIF]
14 RBIF	Receive Message Buffer Interrupt Flag. This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted. 1 At least one individual receive message buffers has the MBIF flag asserted.
15 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted. 1 At least one individual transmit message buffers has the MBIF flag asserted.

60.7.31 System Memory Access Time-Out Register (FR_SYMATOR)

Write: Disabled Mode

Address: 0h base + 3Eh offset = 3Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								TIMEOUT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

FR_SYMATOR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 TIMEOUT	System Memory Access Time-Out. This value defines when a system bus access timout is detected. For a detailed description see Configure System Memory Access Time-Out Register (FR_SYMATOR) and System Bus Access Timeout .

60.7.32 Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 MT after offset correction start.

NOTE

If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

NOTE

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the values SFODB and SFODA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFEVB				SFEVA				SFODB				SFODA			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFCNTR field descriptions

Field	Description
0–3 SFEVB	Sync Frames Channel B, even cycle. protocol related variable: size of (vsSyncIdListB for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4–7 SFEVA	Sync Frames Channel A, even cycle. protocol related variable: size of (vsSyncIdListA for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8–11 SFODB	Sync Frames Channel B, odd cycle. protocol related variable: size of (vsSyncIdListB for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .
12–15 SFODA	Sync Frames Channel A, odd cycle. protocol related variable: size of (vsSyncIdListA for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .

60.7.33 Sync Frame Table Offset Register (FR_SFTOR)

Write: POC:config

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 42h offset = 42h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFT_OFFSET															0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFTOR field descriptions

Field	Description
0–14 SFT_OFFSET	Sync Frame Table Offset. The offset of the Sync Frame Tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus STF_OFFSET[0] is always 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.34 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Write: Normal Mode

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7
Read	0	0	CYCNUM					
Write	ELKT	OLKT						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
Write						OPT		
Reset	0	0	0	0	0	0	0	0

FR_SFTCCSR field descriptions

Field	Description
0 ELKT	Even Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables.
1 OLKT	Odd Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables.

Table continues on the next page...

FR_SFTCCSR field descriptions (continued)

Field	Description
2–7 CYCNUM	Cycle Number. This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	Even Cycle Tables Lock Status. This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables. 1 Application has locked the even cycle tables.
9 OLKS	Odd Cycle Tables Lock Status. This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables. 1 Application has locked the odd cycle tables.
10 EVAL	Even Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
11 OVAL	Odd Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 OPT	One Pair Trigger. This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.
14 SDVEN	Sync Frame Deviation Table Enable. This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. **NOTE:** If SDVEN is set to 1, then SIDEN must also be set to 1. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area
15 SIDEN	Sync Frame ID Table Enable. This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

60.7.35 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Write: Normal Mode

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Address: 0h base + 46h offset = 46h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								SYNFRID							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDRFR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SYNFRID	Sync Frame Rejection ID. This field defines the frame ID of a frame that must not be used for clock synchronization. For details see Sync Frame Rejection Filtering .

60.7.36 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Write: POC:config

This register defines the sync frame acceptance filter value. For details on filtering, see Sync Frame Filtering .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								FVAL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFVR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FVAL	Filter Value. This field defines the value for the sync frame acceptance filtering.

60.7.37 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Write: POC:config

This register defines the sync frame acceptance filter mask. For details on filtering see Sync Frame Acceptance Filtering .

Address: 0h base + 4Ah offset = 4Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						FMSK									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFMR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FMSK	Filter Mask. This field defines the mask for the sync frame acceptance filtering.

60.7.38 Network Management Vector Register (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0-FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Address: 0h base + 4Ch offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	NMVP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVR*n* field descriptions

Field	Description
0–15 NMVP	Network Management Vector Part. The mapping between the Network Management Vector Registers (FR_NMVR0-FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in the following table.

Table 60-10. Mapping of NMVRn to the Received Payload Bytes NMVn

NMVRn Register	NMVn Received Payload
FR_NMVR0[NMVP[15:8]]	NMV0
FR_NMVR0[NMVP[7:0]]	NMV1
FR_NMVR1[NMVP[15:8]]	NMV2
FR_NMVR1[NMVP[7:0]]	NMV3
...	
FR_NMVR5[NMVP[15:8]]	NMV10
FR_NMVR5[NMVP[7:0]]	NMV11

60.7.39 Network Management Vector Length Register (FR_NMVLR)

Write: POC:config

This register defines the length of the network management vector in bytes.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												NMVL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVLR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 NMVL	Network Management Vector Length. protocol related variable: gNetworkManagementVectorLength This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

60.7.40 Timer Configuration and Control Register (FR_TICCR)

Write: T2_CFG: POC:config T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal Mode

This register is used to configure and control the two timers T1 and T2. For timer details, see Timer Support . The Timer T1 is an absolute timer. The Timer T2 can be configured as an absolute or relative timer.

NOTE

Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

Address: 0h base + 5Ah offset = 5Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2_CFG	T2_REP	0	0	0	T2ST	0			T1_REP	0	0	0	T1ST
Write						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TICCR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 T2_CFG	Timer T2 Configuration. This bit configures the timebase mode of Timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	Timer T2 Repetitive Mode. This bit configures the repetition mode of Timer T2. 0 T2 is non repetitive 1 T2 is repetitive
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 T2SP	Timer T2 Stop. This trigger bit is used to stop timer T2. 0 no effect 1 stop timer T2
6 T2TR	Timer T2 Trigger. This trigger bit is used to start timer T2. 0 no effect 1 start timer T2
7 T2ST	Timer T2 State. This status bit provides the current state of timer T2. 0 timer T2 is idle 1 timer T2 is running
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 T1_REP	Timer T1 Repetitive Mode. This bit configures the repetition mode of timer T1. 0 T1 is non repetitive 1 T1 is repetitive

Table continues on the next page...

FR_TICCR field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 T1SP	Timer T1 Stop. This trigger bit is used to stop timer T1. 0 no effect 1 stop timer T1
14 T1TR	Timer T1 Trigger. This trigger bit is used to start timer T1. 0 no effect 1 start timer T1
15 T1ST	Timer T1 State. This status bit provides the current state of timer T1. 0 timer T1 is idle 1 timer T1 is running

60.7.41 Timer 1 Cycle Set Register (FR_TI1CYSR)

Write: Anytime

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_CYC_VAL						0		T1_CYC_MSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1CYSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T1_CYC_VAL	Timer T1 Cycle Filter Value. This field defines the cycle filter value for timer T1.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T1_CYC_MSK	Timer T1 Cycle Filter Mask. This field defines the cycle filter mask for timer T1.

60.7.42 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

Write: Anytime

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Eh offset = 5Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_MTOFFSET													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1MTOR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T1_MTOFFSET	Timer 1 Macrotick Offset. This field defines the macrotick offset value for timer 1.

60.7.43 Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, .

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2CYCVAL						0		T2CYCMSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T2CYCVAL	Timer T2 Cycle Filter Mask
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T2CYCMSK	Timer T2 Cycle Filter Mask

60.7.44 Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	T2MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick High Word

60.7.45 Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Absolute Timer T2

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2MOFF													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T2MOFF	Timer T2 Macrotick Offset

60.7.46 Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	T2MTCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick Low Word

60.7.47 Slot Status Selection Register (FR_SSSR)

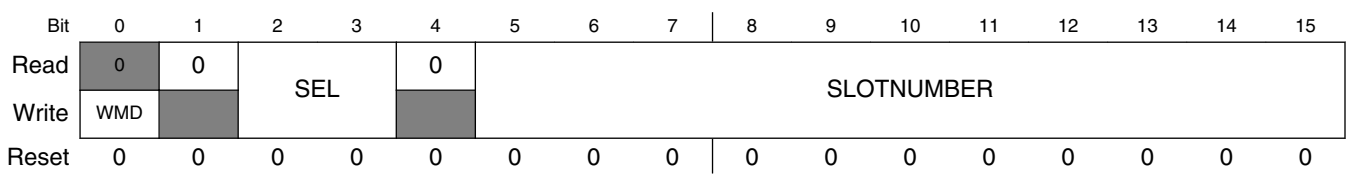
Write: Anytime

This register is used to access the four internal non-memory mapped slot status selection registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0-FR_SSR7) according to Slot Status Selection Register (FR_SSSR) . For a detailed description of slot status monitoring, refer to Slot Status Monitoring .

Table 60-11. Mapping Between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[15:8]	FR_SSR0[7:0]	FR_SSR1[15:8]	FR_SSR1[7:0]
FR_SSSR1	FR_SSR2[15:8]	FR_SSR2[7:0]	FR_SSR3[15:8]	FR_SSR3[7:0]
FR_SSSR2	FR_SSR4[15:8]	FR_SSR4[7:0]	FR_SSR5[15:8]	FR_SSR5[7:0]
FR_SSSR3	FR_SSR6[15:8]	FR_SSR6[7:0]	FR_SSR7[15:8]	FR_SSR7[7:0]

Address: 0h base + 64h offset = 64h



FR_SSSR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_SSSR field descriptions (continued)

Field	Description
2–3 SEL	Selector. This field selects one of the four internal slot status selection registers for access. 00 select FR_SSSR0. 01 select FR_SSSR1. 10 select FR_SSSR2. 11 select FR_SSSR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTNUMBER	Slot Number. This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **NOTE:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

60.7.48 Slot Status Counter Condition Register (FR_SSCCR)

Write: Anytime

This register is used to access and program the four internal non-memory mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal slot status counter condition registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0-FR_SSCR3). The correspondence is given in Slot Status Counter Condition Register (FR_SSCCR) . For a detailed description of slot status counters, refer to Slot Status Counter Registers .

Table 60-12. Mapping between internal FR_SSCCRn and FR_SSCRn

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

Address: 0h base + 66h offset = 66h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SEL	Selector. This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0. 01 select FR_SSCCR1. 10 select FR_SSCCR2. 11 select FR_SSCCR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–6 CNTCFG	Counter Configuration. These bit field controls the channel related incrementing of the slot status counter. 00 increment by 1 if condition is fulfilled on channel A. 01 increment by 1 if condition is fulfilled on channel B. 10 increment by 1 if condition is fulfilled on at least one channel. 11 increment by 2 if condition is fulfilled on both channels channel; increment by 1 if condition is fulfilled on only one channel.
7 MCY	Multi Cycle Selection. This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	Valid Frame Restriction. This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	Sync Frame Restriction. This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	Null Frame Restriction. This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.
11 SUF	Startup Frame Restriction. This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12–15 STATUSMASK	Slot Status Mask. This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3] - This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2] - This bit enables the counting for slots with the content error indicator bit set to 1.

Table continues on the next page...

FR_SSCCR field descriptions (continued)

Field	Description
	STATUSMASK[1] - This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0] - This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

60.7.49 Slot Status Register (FR_SSR*n*)

Each Slot Status Register (SSR) holds the status vector of the slot specified in the corresponding internal slot status selection register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in Figure 60-31 . The register bits are directly related to the protocol variables and described in more detail in Slot Status Monitoring .

Address: 0h base + 68h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSR*n* field descriptions

Field	Description
0 VFB	Valid Frame on Channel B. protocol related variable: vSS!ValidFrame channel B 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
1 SYB	Sync Frame Indicator Channel B. protocol related variable: vRF!Header!SyFIndicator channel B 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
2 NFB	Null Frame Indicator Channel B. protocol related variable: vRF!Header!NFIndicator channel B 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
3 SUB	Startup Frame Indicator Channel B. protocol related variable: vRF!Header!SuFIndicator channel B 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
4 SEB	Syntax Error on Channel B. protocol related variable: vSS!SyntaxError channel B 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1

Table continues on the next page...

FR_SSR*n* field descriptions (continued)

Field	Description
5 CEB	Content Error on Channel B. protocol related variable: vSS!ContentError channel B 0 vSS!ContentError = 0 1 vSS!ContentError = 1
6 BVB	Boundary Violation on Channel B. protocol related variable: vSS!BViolation channel B 0 vSS!BViolation = 0 1 vSS!BViolation = 1
7 TCB	Transmission Conflict on Channel B. protocol related variable: vSS!TxConflict channel B 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1
8 VFA	Valid Frame on Channel A. protocol related variable: vSS!ValidFrame channel A 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
9 SYA	Sync Frame Indicator Channel A. protocol related variable: vRF!Header!SyFIndicator channel A 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
10 NFA	Null Frame Indicator Channel A. protocol related variable: vRF!Header!NFIndicator channel A 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
11 SUA	Startup Frame Indicator Channel A. protocol related variable: vRF!Header!SuFIndicator channel A 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
12 SEA	Syntax Error on Channel A. protocol related variable: vSS!SyntaxError channel A 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1
13 CEA	Content Error on Channel A. protocol related variable: vSS!ContentError channel A 0 vSS!ContentError = 0 1 vSS!ContentError = 1
14 BVA	Boundary Violation on Channel A. protocol related variable: vSS!BViolation channel A 0 vSS!BViolation = 0 1 vSS!BViolation = 1
15 TCA	Transmission Conflict on Channel A. protocol related variable: vSS!TxConflict channel A 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1

60.7.50 Slot Status Counter Register (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal slot status counter condition register FR_SSCCRn, which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, see Slot Status Counter Registers .

NOTE

If the counter has reached its maximum value 0xFFFF and is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be set into the multicycle mode again.

NOTE

Address: 0h base + 78h offset + (2d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SLOTSTATUSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCR*n* field descriptions

Field	Description
0–15 SLOTSTATUSCNT	Slot Status Counter. This field provides the current value of the Slot Status Counter.

60.7.51 MTS A Configuration Register (FR_MTSACFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel A. For more details, see MTS Generation .

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSACFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

60.7.52 MTS B Configuration Register (FR_MTSBCFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel B. For more details, see MTS Generation .

Address: 0h base + 82h offset = 82h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSBCFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

60.7.53 Receive Shadow Buffer Index Register (FR_RSBIR)

Write: WMD, SEL: Any Time; RSBIDX: POC:config

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers . For more details on the receive shadow buffer concept, refer to Receive Shadow Buffers Concept .

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7
Read Write	WMD	Reserved	SEL		Reserved			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	RSBIDX							
Reset	0	0	0	0	0	0	0	0

FR_RSBIR field descriptions

Field	Description
0 WMD	Write Mode. This bit controls the write mode for this register. 0 update SEL and RSBIDX field on register write 1 update only SEL field on register write
1 Reserved	This field is reserved.
2–3 SEL	Selector. This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1. receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2. receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1. receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2. receive shadow buffer index register for channel B, segment 2
4–7 Reserved	This field is reserved.
8–15 RSBIDX	RSBIDXA1/RSBIDXA2/RSBIDXB1/RSBIDXB2- Receive Shadow Buffer Index Receive Shadow Buffer Index. This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

60.7.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Write: WMA/WMB: POC:config, SEL: Anytime

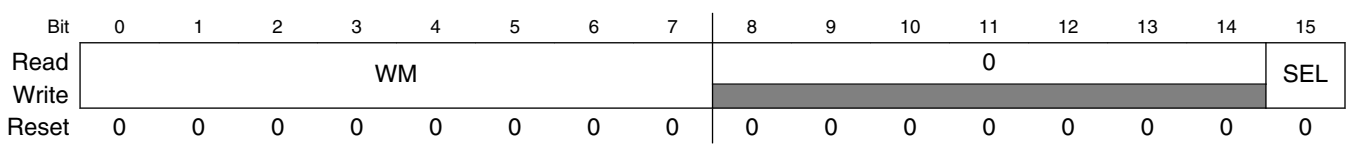
This register is used to

- select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in Receive FIFO Watermark and Selection Register (FR_RFWMSR) .
- to define the watermark for the selected FIFO.

Table 60-13. SEL Controlled Receiver FIFO Registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Address: 0h base + 86h offset = 86h



FR_RFWMSR field descriptions

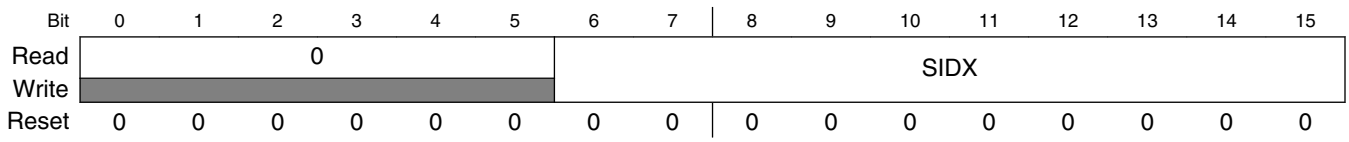
Field	Description
0–7 WM	WMA/WMB - Watermark This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 SEL	Select. This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

60.7.55 Receive FIFO Start Index Register (FR_RFSIR)

Write: POC:config

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Address: 0h base + 88h offset = 88h



FR_RFSIR field descriptions

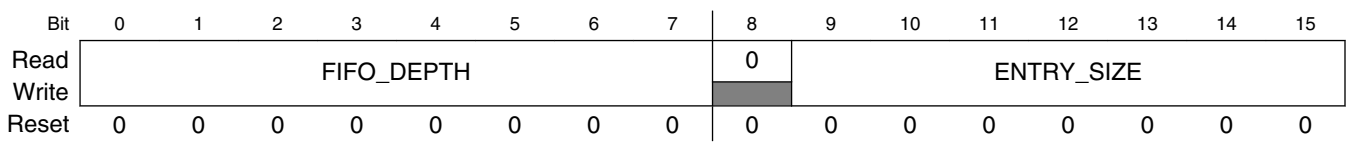
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SIDX	SIDXA/SIDXB - Start Index This field defines the number of the message buffer header field of the first message buffer of the selected FIFO. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

60.7.56 Receive FIFO Depth and Size Register (FR_RFDSR)

Write: POC:config

This register defines the structure of the selected FIFO, i.e. the number of entries and the size of each entry .

Address: 0h base + 8Ah offset = 8Ah



FR_RFDSR field descriptions

Field	Description
0–7 FIFO_DEPTH	FIFO_DEPTHA/FIFO_DEPTHB - FIFO Depth FIFO Depth. This field defines the depth of the selected FIFO, i.e. the number of entries. Note: If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 too, to ensure that no frames are received into the FIFO.

Table continues on the next page...

FR_RFDSR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ENTRY_SIZE	ENTRY_SIZEA/ENTRY_SIZEB - Entry Size This field defines the size of the frame data sections for the selected FIFO in 2 byte entities.

60.7.57 Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						RDIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFARIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

60.7.58 Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Eh offset = 8Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								RDIDX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFBRIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

60.7.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Write: POC:config

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFVR field descriptions

Field	Description
0–15 MIDAFVAL	MIDAFVALA/MIDAFVALB - Message ID Acceptance Filter Value Filter value for the message ID acceptance filter.

60.7.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Write: POC:config

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 92h offset = 92h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFMSK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFMR field descriptions

Field	Description
0–15 MIDAFMSK	MIDAFMSKA/MIDAFMSKB - Message ID Acceptance Filter Mask Filter mask for the message ID acceptance filter.

60.7.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Write: POC:config

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFVAL										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFVR field descriptions

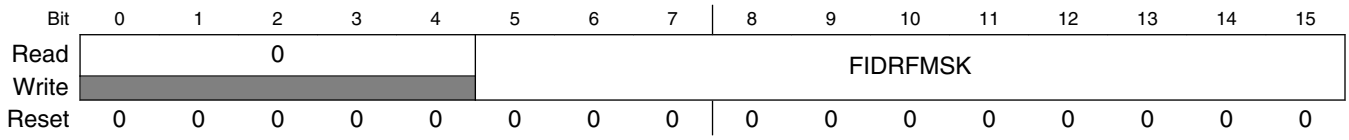
Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFVAL	FIDRFVALA/FIDRFVALB - Frame ID Rejection Filter Value Filter value for the frame ID rejection filter.

60.7.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Write: POC:config

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 96h offset = 96h



FR_RFFIDRFMR field descriptions

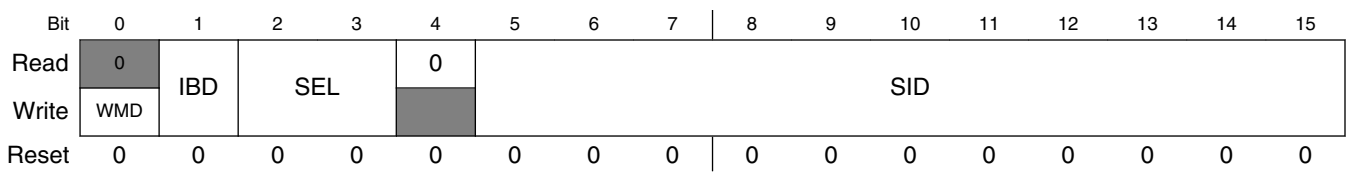
Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFMSK	Frame ID Rejection Filter Mask. Filter mask for the frame ID rejection filter.

60.7.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Write: WMD, IBD, SEL: Any Time; SID: POC:config

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter see FIFO Filtering .

Address: 0h base + 98h offset = 98h



FR_RFRFCFR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL and IBD field only on write access.
1 IBD	Interval Boundary. This control bit selects the interval boundary to be programmed with the SID value. 0 program lower interval boundary 1 program upper interval boundary
2–3 SEL	Filter Selector. This control field selects the frame ID range filter to be accessed. 00 select frame ID range filter 0. 01 select frame ID range filter 1. 10 select frame ID range filter 2. 11 select frame ID range filter 3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCFR field descriptions (continued)

Field	Description
5–15 SID	Slot ID Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

60.7.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Write: Anytime

This register is used to enable and disable each frame ID range filter and to define whether it is running as acceptance or rejection filter.

Address: 0h base + 9Ah offset = 9Ah

Bit	0	1	2	3	4	5	6	7
Read	0				F3MD	F2MD	F1MD	F0MD
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				F3EN	F2EN	F1EN	F0EN
Write								
Reset	0	0	0	0	0	0	0	0

FR_RFRFCTR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 F3MD	Range Filter 3 Mode. This control bit defines the filter mode of the frame ID range filter 3. 0 range filter 3 runs as acceptance filter 1 range filter 3 runs as rejection filter
5 F2MD	Range Filter 2 Mode. This control bit defines the filter mode of the frame ID range filter 2. 0 range filter 2 runs as acceptance filter 1 range filter 2 runs as rejection filter
6 F1MD	Range Filter 1 Mode. This control bit defines the filter mode of the frame ID range filter 1. 0 range filter 1 runs as acceptance filter 1 range filter 1 runs as rejection filter
7 F0MD	Range Filter 0 Mode. This control bit defines the filter mode of the frame ID range filter 0. 0 range filter 0 runs as acceptance filter 1 range filter 0 runs as rejection filter
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCTR field descriptions (continued)

Field	Description
12 F3EN	Range Filter 3 Enable. This control bit is used to enable and disable the frame ID range filter 3. 0 range filter 3 disabled 1 range filter 3 enabled
13 F2EN	Range Filter 2 Enable. This control bit is used to enable and disable the frame ID range filter 2. 0 range filter 2 disabled 1 range filter 2 enabled
14 F1EN	Range Filter 1 Enable. This control bit is used to enable and disable the frame ID range filter 1. 0 range filter 1 disabled 1 range filter 1 enabled
15 F0EN	Range Filter 0 Enable. This control bit is used to enable and disable the frame ID range filter 0. 0 range filter 0 disabled 1 range filter 0 enabled

60.7.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTA	Last Dynamic Transmission Slot Channel A. protocol related variable: zLastDynTxSlot channel A Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

60.7.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Eh offset = 9Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTB	Last Dynamic Transmission Slot Channel B. protocol related variable: zLastDynTxSlot channel B Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

60.7.67 Protocol Configuration Register 0 (FR_PCR0)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Table 60-14. Protocol Configuration Register (PCR0-30) Fields

Name	Description	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts			number	3
action_point_offset	gdActionPointOffset - 1			MT	0
cas_rx_low_max	gdCASRxLowMax - 1			gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase			minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1			MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1			MT	2
static_slot_length	gdStaticSlot			MT	0

Table continues on the next page...

Table 60-14. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1			MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1			MT	6
tss_transmitter	gdTSSTransmitter			gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle			gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow			gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow			gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle			gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow			gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1			µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]			MT	6
macro_initial_offset_b	pMacroInitialOffset[B]			MT	16
macro_per_cycle	gMacroPerCycle			MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot			MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart			MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal			cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive			cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1			minislot	29
number_of_static_slots	gNumberOfStaticSlots			static slot	2
offset_correction_start	gOffsetCorrectionStart			MT	11
payload_length_static	gPayloadLengthStatic			2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax			2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1			MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock			bool	26
allow_passive_to_active	pAllowPassiveToActive			cyclepairs	12
cluster_drift_damping	pClusterDriftDamping			µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]			µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]			µT	26
listen_timeout	pdListenTimeout - 1			µT	14/15
key_slot_id	pKeySlotId			number	18
key_slot_used_for_startup	pKeySlotUsedForStartup			bool	11
key_slot_used_for_sync	pKeySlotUsedForSync			bool	11
latest_tx	gNumberOfMinislots - pLatestTx			minislot	21
sync_node_max	gSyncNodeMax			number	30
micro_initial_offset_a	pMicroInitialOffset[A]			µT	20

Table continues on the next page...

Table 60-14. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
micro_initial_offset_b	pMicroInitialOffset[B]			µT	20
micro_per_cycle	pMicroPerCycle			µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift			µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift			µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)			µT	7
offset_correction_out	pOffsetCorrectionOut			µT	9
rate_correction_out	pRateCorrectionOut			µT	14
single_slot_enabled	pSingleSlotEnabled			bool	10
wakeup_channel	pWakeupChannel	see Protocol Configuration Register 0 (FR_PCR0)			10
wakeup_pattern	pWakeupPattern			number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2			µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2			µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection			µT	29
extern_rate_correction	pExternRateCorrection			µT	21

Table 60-15. Wakeup Channel Selection

wakeup_channel	Wakeup Channel
0	A
1	B

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	action_point_offset								static_slot_length							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR0 field descriptions

Field	Description
0–5 action_point_offset	gdActionPointOffset - 1
6–15 static_slot_length	gdStaticSlot

60.7.68 Protocol Configuration Register 1 (FR_PCR1)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A2h offset = A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		macro_after_first_static_slot													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 macro_after_ first_static_slot	gMacroPerCycle - gdStaticSlot

60.7.69 Protocol Configuration Register 2 (FR_PCR2)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	minislot_after_action_point						number_of_static_slots									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR2 field descriptions

Field	Description
0–5 minislot_after_ action_point	gdMinislot - gdMinislotActionPointOffset - 1
6–15 number_of_ static_slots	gNumberOfStaticSlots

60.7.70 Protocol Configuration Register 3 (FR_PCR3)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A6h offset = A6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_symbol_rx_low						minislot_action_point_offset					coldstart_attempts				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR3 field descriptions

Field	Description
0–5 wakeup_symbol_rx_low	gdWakeupSymbolRxLow
6–10 minislot_action_point_offset	gdMinislotActionPointOffset - 1
11–15 coldstart_attempts	gColdstartAttempts

60.7.71 Protocol Configuration Register 4 (FR_PCR4)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cas_rx_low_max							wakeup_symbol_rx_window								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR4 field descriptions

Field	Description
0–6 cas_rx_low_max	gdCASRxLowMax - 1

Table continues on the next page...

FR_PCR4 field descriptions (continued)

Field	Description
7–15 wakeup_symbol_ rx_window	gdWakeupSymbolRxWindow

60.7.72 Protocol Configuration Register 5 (FR_PCR5)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AAh offset = AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	tss_transmitter				wakeup_symbol_tx_low						wakeup_symbol_rx_idle					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR5 field descriptions

Field	Description
0–3 tss_transmitter	gdTSSTransmitter
4–9 wakeup_symbol_ tx_low	gdWakeupSymbolTxLow
10–15 wakeup_symbol_ rx_idle	gdWakeupSymbolRxIdle

60.7.73 Protocol Configuration Register 6 (FR_PCR6)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	symbol_window_after_action_point							macro_initial_offset_a							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR6 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–8 symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1
9–15 macro_initial_offset_a	pMacroInitialOffset[A]

60.7.74 Protocol Configuration Register 7 (FR_PCR7)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AEh offset = AEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_b								micro_per_macro_nom_half							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR7 field descriptions

Field	Description
0–8 decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2
9–15 micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)

60.7.75 Protocol Configuration Register 8 (FR_PCR8)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	max_without_clock_correction_fatal				max_without_clock_correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR8 field descriptions

Field	Description
0–3 max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal
4–7 max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive
8–15 wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle

60.7.76 Protocol Configuration Register 9 (FR_PCR9)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B2h offset = B2h

Bit	0	1	2	3	4	5	6	7
Read Write	minislot_exists	symbol_window_exists	offset_correction_out					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_out							
Reset	0	0	0	0	0	0	0	0

FR_PCR9 field descriptions

Field	Description
0 minislot_exists	gNumberOfMinislots!=0
1 symbol_window_exists	gdSymbolWindow!=0

Table continues on the next page...

FR_PCR9 field descriptions (continued)

Field	Description
2–15 offset_ correction_out	pOffsetCorrectionOut

60.7.77 Protocol Configuration Register 10 (FR_PCR10)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7
Read Write	single_slot_ enabled	wakeup_ channel	macro_per_cycle					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	macro_per_cycle							
Reset	0	0	0	0	0	0	0	0

FR_PCR10 field descriptions

Field	Description
0 single_slot_ enabled	pSingleSlotEnabled
1 wakeup_channel	pWakeupChannel
2–15 macro_per_cycle	gMacroPerCycle

60.7.78 Protocol Configuration Register 11 (FR_PCR11)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B6h offset = B6h

Bit	0	1	2	3	4	5	6	7
Read Write	key_slot_used_for_startup	key_slot_used_for_sync	offset_correction_start					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_start							
Reset	0	0	0	0	0	0	0	0

FR_PCR11 field descriptions

Field	Description
0 key_slot_used_for_startup	pKeySlotUsedForStartup
1 key_slot_used_for_sync	pKeySlotUsedForSync
2–15 offset_correction_start	gOffsetCorrectionStart

60.7.79 Protocol Configuration Register 12 (FR_PCR12)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	allow_passive_to_active					key_slot_header_crc										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR12 field descriptions

Field	Description
0–4 allow_passive_to_active	pAllowPassiveToActive
5–15 key_slot_header_crc	header CRC for key slot

60.7.80 Protocol Configuration Register 13 (FR_PCR13)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BAh offset = BAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	first_minislot_action_point_offset						static_slot_after_action_point									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR13 field descriptions

Field	Description
0–5 first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1
6–15 static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1

60.7.81 Protocol Configuration Register 14 (FR_PCR14)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	rate_correction_out											listen_timeout				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR14 field descriptions

Field	Description
0–10 rate_correction_out	pRateCorrectionOut
11–15 listen_timeout	pdListenTimeout - 1

60.7.82 Protocol Configuration Register 15 (FR_PCR15)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BEh offset = BEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR15 field descriptions

Field	Description
0–15 listen_timeout	pdListenTimeout - 1

60.7.83 Protocol Configuration Register 16 (FR_PCR16)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	macro_initial_offset_b							noise_listen_timeout								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR16 field descriptions

Field	Description
0–6 macro_initial_ offset_b	pMacroInitialOffset[B]
7–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

60.7.84 Protocol Configuration Register 17 (FR_PCR17)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	noise_listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR17 field descriptions

Field	Description
0–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

60.7.85 Protocol Configuration Register 18 (FR_PCR18)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_pattern						key_slot_id									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR18 field descriptions

Field	Description
0–5 wakeup_pattern	pWakeupPattern
6–15 key_slot_id	pKeySlotId

60.7.86 Protocol Configuration Register 19 (FR_PCR19)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_a								payload_length_static							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR19 field descriptions

Field	Description
0–8 decoding_ correction_a	pDecodingCorrection + pDelayCompensation[A] + 2
9–15 payload_length_ static	gPayloadLengthStatic

60.7.87 Protocol Configuration Register 20 (FR_PCR20)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR20 field descriptions

Field	Description
0–7 micro_initial_ offset_b	pMicroInitialOffset[A]
8–15 micro_initial_ offset_a	pMicroInitialOffset[A]

60.7.88 Protocol Configuration Register 21 (FR_PCR21)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_rate_ correction			latest_tx												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR21 field descriptions

Field	Description
0–2 extern_rate_ correction	pExternRateCorrection
3–15 latest_tx	gNumberOfMinislots - pLatestTx

60.7.89 Protocol Configuration Register 22 (FR_PCR22)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved	comp_accepted_startup_range_a						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_a				micro_per_cycle			
Reset	0	0	0	0	0	0	0	0

FR_PCR22 field descriptions

Field	Description
0 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value.

Table continues on the next page...

FR_PCR22 field descriptions (continued)

Field	Description
	This field is reserved.
1–11 comp_accepted_ startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]
12–15 micro_per_cycle	pMicroPerCycle

60.7.90 Protocol Configuration Register 23 (FR_PCR23)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR23 field descriptions

Field	Description
0–15 micro_per_cycle	pMicroPerCycle

60.7.91 Protocol Configuration Register 24 (FR_PCR24)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cluster_drift_damping					max_payload_length_dynamic							micro_per_cycle_min			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR24 field descriptions

Field	Description
0–4 cluster_drift_ damping	pClusterDriftDamping
5–11 max_payload_ length_dynamic	pPayloadLengthDynMax
12–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

60.7.92 Protocol Configuration Register 25 (FR_PCR25)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D2h offset = D2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_min															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR25 field descriptions

Field	Description
0–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

60.7.93 Protocol Configuration Register 26 (FR_PCR26)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7
Read Write	allow_halt_due_to_clock	comp_accepted_startup_range_b						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_b				micro_per_cycle_max			
Reset	0	0	0	0	0	0	0	0

FR_PCR26 field descriptions

Field	Description
0 allow_halt_due_to_clock	pAllowHaltDueToClock
1–11 comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]
12–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

60.7.94 Protocol Configuration Register 27 (FR_PCR27)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D6h offset = D6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_max															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR27 field descriptions

Field	Description
0–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

60.7.95 Protocol Configuration Register 28 (FR_PCR28)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	dynamic_slot_idle_phase		macro_after_offset_correction													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR28 field descriptions

Field	Description
0–1 dynamic_slot_idle_phase	gdDynamicSlotIdlePhase.
2–15 macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart

60.7.96 Protocol Configuration Register 29 (FR_PCR29)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DAh offset = DAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	extern_offset_correction			minislots_max												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR29 field descriptions

Field	Description
0–2 extern_offset_correction	pExternOffsetCorrection

Table continues on the next page...

FR_PCR29 field descriptions (continued)

Field	Description
3–15 minislots_max	gNumberOfMinislots - 1

60.7.97 Protocol Configuration Register 30 (FR_PCR30)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												sync_node_max			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR30 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 sync_node_max	gSyncNodeMax

60.7.98 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Write: POC:config

NOTE

Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses the content of the FR_RFSDOR corresponds to the start address of payload area of the selected FIFO.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSDOR field descriptions

Field	Description
0–15 SDO	SDOA/SDOB - Start Data Field Offset Start Data Field Offset. This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO's first message buffer header field. For configuration constraints see Configure Data Field Offsets .

60.7.99 Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

60.7.100 Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + EAh offset = EAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SMBA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

60.7.101 Receive FIFO Periodic Timer Register (FR_RFPTR)

Write: POC:config

This register holds periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see FIFO Periodic Timer).

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								PTD							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFPTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 PTD	Periodic Timer Duration. This value defines the periodic timer duration in terms of macroticks. 0000 timer stays expired 3FFF timer never expires other timer expires after specified number of macroticks, expires and is restarted at each cycle start

60.7.102 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs

If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, than the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

.

Address: 0h base + EEh offset = EEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLB_or_PCB								FLA_or_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFLPCR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–7
FLB_or_PCB</td><td>NOTE: The name and function of the fields in this register vary depending on whether they are being read or written. See the following table for details.

<table>
<tr><th>Operation</th><th>Sub-Field</th><th>Description</th></tr>
<tr><td rowspan="2">Read</td><td>0-7 FLB</td><td>Fill Level FIFO B. This field provides the current number of entries in the FIFO B.</td></tr>
<tr><td>-</td><td>-</td></tr>
<tr><td rowspan="2">Write</td><td>0-7 PCB</td><td>Pop Count FIFO B. This field defines the number of entries to be removed from FIFO B.</td></tr>
<tr><td>-</td><td>-</td></tr>
</table></td></tr>
<tr><td>8–15
FLA_or_PCA</td><td>
<table>
<tr><th>Operation</th><th>Sub-Field</th><th>Description</th></tr>
<tr><td rowspan="2">Read</td><td>-</td><td>-</td></tr>
<tr><td>8-15 FLA</td><td>Fill Level FIFO A- This field provides the current number of entries in the FIFO A.</td></tr>
<tr><td rowspan="2">Write</td><td>-</td><td>-</td></tr>
<tr><td>8-15 PCA</td><td>Pop Count FIFO A - This field defines the number of entries to be removed from FIFO A.</td></tr>
</table></td></tr>
</table>

60.7.103 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description see Memory Error Reporting .

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7
Read	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
Write								
Reset	0	0	0	0	0	0	0	0

FR_EEIFER field descriptions

Field	Description
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on CHI LRAM and interrupt flag LRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of CHI LRAM 0 no such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears a) memory errors are detected and corrected on CHI LRAM and interrupt flag LRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of CHI LRAM **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error overflow detected on CHI LRAM
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on PE DRAM and interrupt flag DRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of the PE DRAM 0 no such event 1 Non-Corrected Error overflow detected on PE DRAM

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
3 DRCE_OF	DRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected and corrected on PE DRAM and interrupt flag DRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of PE DRAM 0 no such event 1 Corrected Error overflow detected on PE DRAM
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on the CHI LRAM. 0 no such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on the CHI LRAM. **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error detected on CHI LRAM
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on PE DRAM. 0 no such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on PE DRAM. 0 no such event 1 Corrected Error detected on PE DRAM
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 LRNE_IE	LRAM Non-Corrected Error Interrupt Enable. This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 LRCE_IE	LRAM Corrected Error Interrupt Enable. This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	DRAM Non-Corrected Error Interrupt Enable. This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	DRAM Corrected Error Interrupt Enable. This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set.

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
	0 Disable interrupt line 1 Enable interrupt line

60.7.104 ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Address: 0h base + F2h offset = F2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	BSY	0					ERS		0			ERM	0		EIM	EIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy- This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in " Register Write Access 0 ECC configuration is idle 1 ECC configuration is running
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 ERS	Error Report Select. This field selects the content of the ECC Error reporting registers. 00 show PE DRAM non-corrected error information 01 show PE DRAM corrected error information 10 show CHI LRAM non-corrected error information 11 show CHI LRAM corrected error information
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERM	Error Report Mode. This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 store data and code as delivered by ecc decoding logic. 1 store data and code as read from the memory.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 EIM	Error Injection Mode. This bit configures the ECC error injection mode.

Table continues on the next page...

FR_EERICR field descriptions (continued)

Field	Description
	0 use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection. 1 use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection.
15 EIE	Error Injection Enable. This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

60.7.105 ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MID	BANK			ADDR											
Write																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field provides the BANK for which the memory error is reported. 111 reset value, indicates no error found after reset. For MID=0: 000 PE DRAM [7:0] 001 PE DRAM [15:8] others - not used For MID=1: Refer to the following table for the assignment of the LRAM banks.

Table 60-16. LRAM Bank Value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3

Table continues on the next page...

FR_EERAR field descriptions (continued)

Field	Description
	Table 60-16. LRAM Bank Value for MID = 1 (continued)

BANK	Register		
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

4–15 ADDR	Memory Address. This field provides the address of the failing memory location.

60.7.106 ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

Table 60-17. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2									FR_MBIDXR(2n)[MBIDX]							
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5									FR_MBIDXR(2n+1)[MBIDX]							
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

Address: 0h base + F6h offset = F6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Ecc Data, shows data as generated by the ecc decoding logic. ERM=1: Memory Data, shows data as read from the memory.

60.7.107 ECC Error Report Code Register (FR_EERCR)

This register provides the ECC related information of the reported memory read access.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERCR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Syndrome. Shows the ecc syndrome generated by the ecc decoding logic. The coding of the PE DRAM syndrome is shown in PE DRAM Syndrome . The coding of the CHI LRAM syndrome is shown in CHI LRAM Syndrome . ERM=1: Checkbits. Shows the ecc checkbits read from the memory.

60.7.108 ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MID	BANK			ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field defines the memory bank for ECC error injection. For MID=0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] others reserved For MID=1: Refer to ECC Error Report Address Register (FR_EERAR) for the assignment of the LRAM banks.
4–15 ADDR	Memory Address. This flag defines the memory address for ECC error injection.

60.7.109 ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

NOTE

The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to Memory Error Response for details.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the data written into the memory. EIM=1: This field defines the data to be written into the memory.

60.7.110 ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Address: 0h base + FEh offset = FEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEICR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the ecc checkbits written into the memory. EIM=1: This field defines the ecc checkbits written into the memory.

60.7.111 Message Buffer Configuration, Control, Status Register (FR_MBCCSR*n*)

Write: MTD: POC:config or MB_DIS CMT: MB_LCK or MB_DIS EDT, LCKT, MBIE, MBIF: Normal Mode

The content of these registers is composed of message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in Individual Message Buffer Functional Description.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Address: 0h base + 800h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read	0			MTD	CMT		0	MBIE
Write						EDT	LCKT	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DUP	DVAL	EDS	LCKS	MBIF
Write								w1c
Reset	0	0	0	0	0	0	0	0

FR_MBCCSR*n* field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MTD	Message Buffer Transfer Direction. This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
4 CMT	Commit for Transmission. This bit indicates if the transmit message buffer data are ready for transmission. **NOTE:** This bit is read/write but may be modified by hardware other than by a reset. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger. If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger. If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable. This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_MBCCSR*n* field descriptions (continued)

Field	Description
11 DUP	Data Updated. This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and Message buffer data field not updated 1 Frame Header and Message buffer data field updated
12 DVAL	Data Valid. For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 receive message buffer contains no valid frame data / message is transmitted for the first time 1 receive message buffer contains valid frame data / message will be transferred again
13 EDS	Enable/Disable Status. This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled. 1 Message buffer is enabled.
14 LCKS	Lock Status. This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application. 1 Message buffer is locked by the application.
15 MBIF	Message Buffer Interrupt Flag. This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled

60.7.112 Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to Message Buffer Cycle Counter Filtering .

NOTE

If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the

message buffer configuration is illegal and the result of the message buffer search is not defined.

Table 60-18. Channel Assignment Description

CHA	CHB	Transmit Message Buffer		Receive Message Buffer	
		static segment	dynamic segment	static segment	dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Address: 0h base + 802h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read Write	MTM	CHA	CHB	CCFE	CCFMSK			
Reset	0*	0*	0*	0*	0*	0*	0*	0*

Bit	8	9	10	11	12	13	14	15
Read Write	CCFMSK		CCFVAL					
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode. This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event transmission mode 1 State transmission mode
1 CHA	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
2 CHB	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
3 CCFE	Cycle Counter Filtering Enable. This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled

Table continues on the next page...

FR_MBCCFR*n* field descriptions (continued)

Field	Description
4–9 CCFMSK	Cycle Counter Filtering Mask. This field defines the filter mask for the cycle counter filtering.
10–15 CCFVAL	Cycle Counter Filtering Value. This field defines the filter value for the cycle counter filtering.

60.7.113 Message Buffer Frame ID Register (FR_MBFIDR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 804h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FID										
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBFIDR*n* field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FID	Frame ID. The semantic of this field depends on the message buffer transfer type. • Receive Message Buffer: This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. • Transmit Message Buffer: This field is used to determine the slot in which the message in this message buffer should be transmitted.

60.7.114 Message Buffer Index Register (FR_MBIDXR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 806h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MBIDX							
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBIDXR*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 MBIDX	Message Buffer Index. This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

60.7.115 Message Buffer Data Field Offset Register (FR_MBDOR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config

Address: 0h base + 1000h offset + (2d × i), where i=0d to 131d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MBDO															
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBDOR*n* field descriptions

Field	Description
0–15 MBDO	Message Buffer Data Field Offset. This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see Configure Data Field Offsets .

60.7.116 LRAM ECC Error Test Register (FR_LEETR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 1108h offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LEETD															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_LEETR*n* field descriptions

Field	Description
0–15 LEETD	LRAM ECC Error Test Data. This field contains the LRAM data belonging to the test register located in LRAM Bank n.

60.8 Functional Description

This section provides a detailed description of the functionality implemented in the CC.

60.8.1 Message Buffer Concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts, the *message buffer control data* and the *physical message buffer*. The message buffer control data are located in dedicated registers. The structure of the message buffer control data depends on the message buffer type and is described in the Message Buffer Types section. The physical message buffer is located in the FlexRay memory area and is described in the next section.

60.8.2 Physical Message Buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in the figure below.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

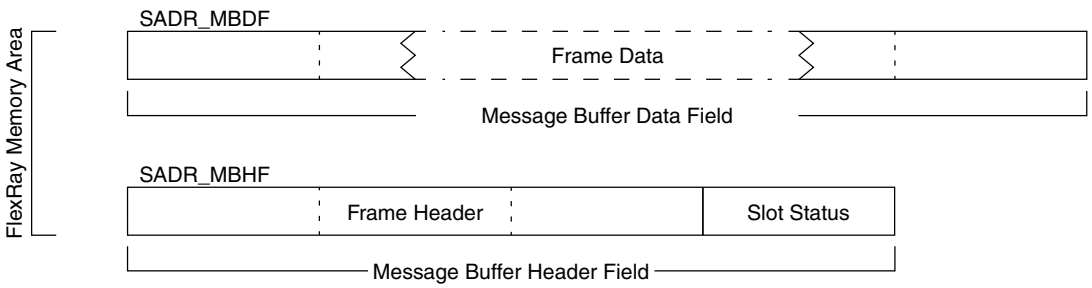


Figure 60-2. Physical Message Buffer Structure

60.8.2.1 Message Buffer Header Field

The message buffer header field is a contiguous region in the FlexRay memory area and occupies eight bytes. It contains the frame header, and the slot status. Its structure is shown in the above figure. The physical start address *SADR_MBHF* of the message buffer header field must be 16-bit aligned.

60.8.2.1.1 Frame Header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header related information according to the *FlexRay Communications System Protocol Specification, Version*2.1 Rev A. A detailed description of the usage and the content of the frame header is provided in Frame Header Description section.

60.8.2.1.2 Slot Status

The slot status occupies the last two bytes of the message buffer header field. It provides the slot and frame status related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in the Slot Status Description section.

60.8.2.2 Message Buffer Data Field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in the following section.

60.8.3 Message Buffer Types

The CC provides three different types of message buffers.

- *Individual Message Buffers*
- *Receive Shadow Buffers*
- *Receive FIFO Buffers*

For each message buffer type the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

60.8.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in the Individual Message Buffer Functional Description section.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in the figure below.

Each individual message buffer has a message buffer number n assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field MBIDX in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR\_MBHF = (FR\_MBIDXRn[MBIDX] \times 8) + SMBA$$

Equation 14

The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to this equation:

$$SADR\_MBDF = [DataFieldOffset] + SMBA$$

Equation 15

The FR_MBDORn are stored in the module internal memory LRAM. Refer to CHI LRAM Initialization for the setup of the data field offset values.

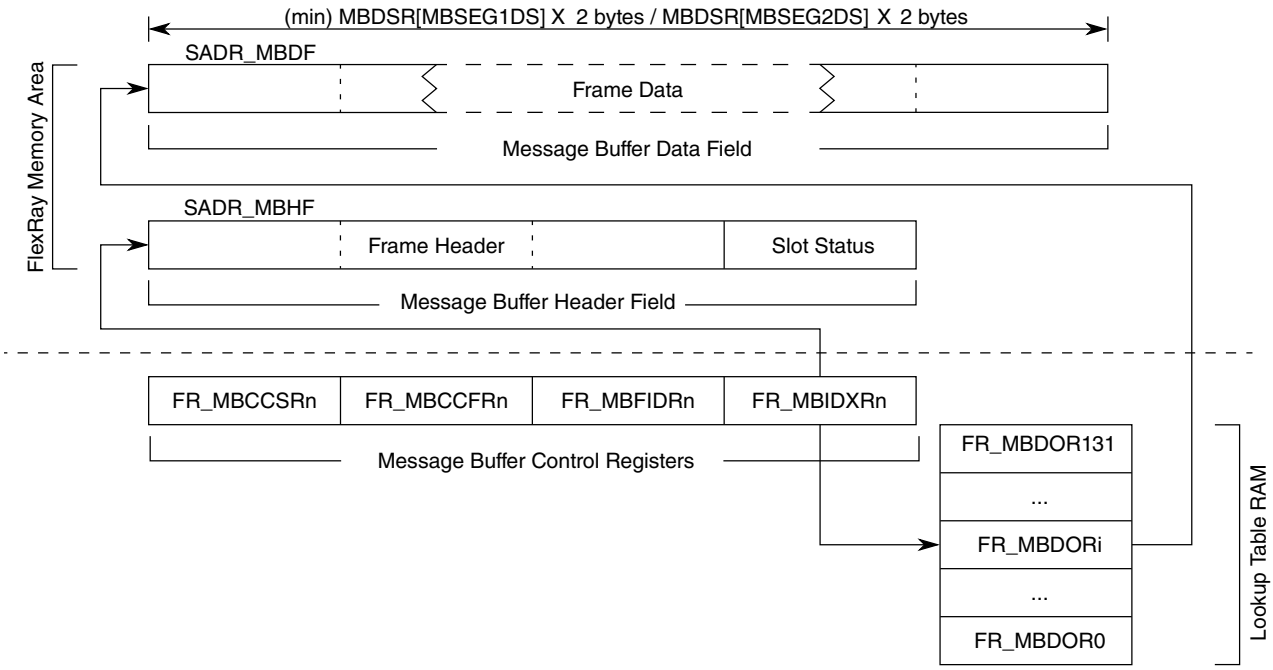


Figure 60-3. Individual Message Buffer Structure

60.8.3.1.1 Individual Message Buffer Segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- *all physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length*
- *the minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes*
- *the minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.*

60.8.3.2 Receive Shadow Buffers

The receive shadow buffers are required for the frame reception process for individual message buffers. The CC provides four receive shadow buffers, one receive shadow buffer per channel and per message buffer segment.

Each receive shadow buffer consists of two parts, the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in the figure below. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR\_MBHF = (FR\_RSBIR[RSBIDX] \times 8) + SMBA$$

Equation 16

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

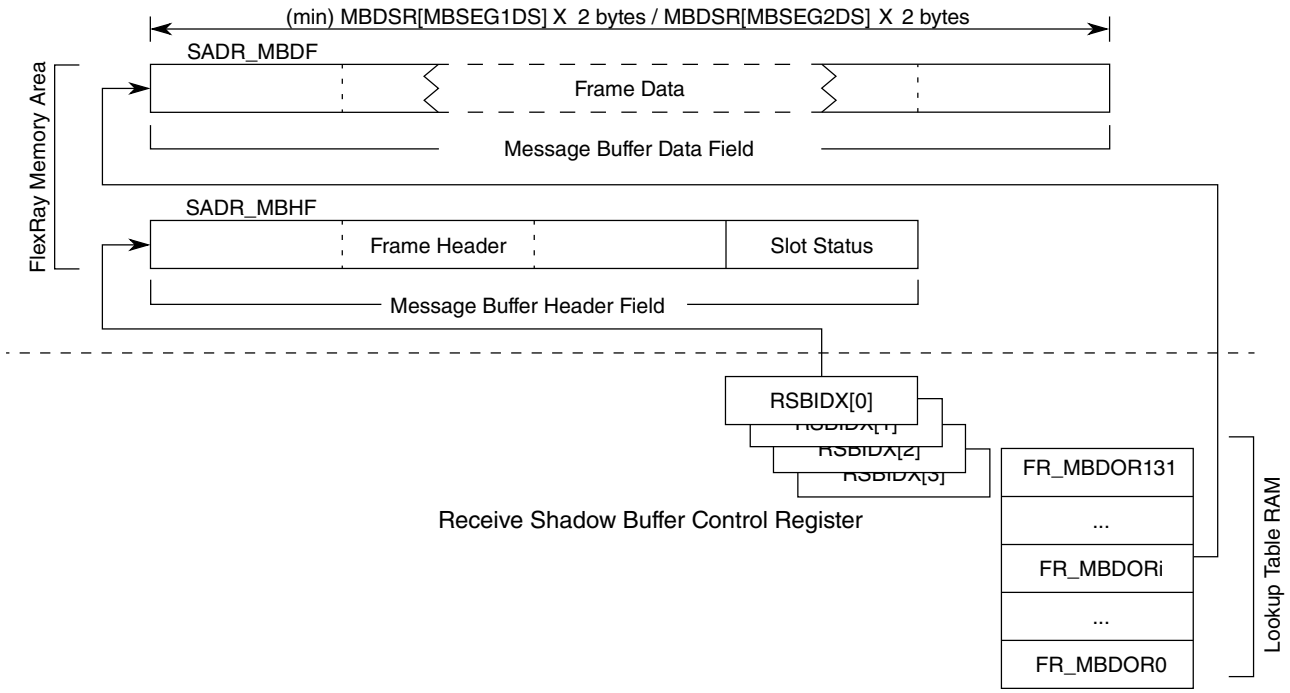


Figure 60-4. Receive Shadow Buffer Structure

60.8.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in the Receive FIFO Structure figure, later in this section.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR) / Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM] in the Module Configuration Register.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to Equation 17 on page 3040.

$$SADR\_MBHF[1] = (8 \times FR\_RFSIR[SIDX]) + SMBA$$

Equation 17

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according Equation 18 on page 3041 .

$$SADR\_MBHF[n] = (8 \times (FR\_RFSIR[SIDX] + FR\_RFDSR[FIFO\_DEPTH])) + SMBA$$

Equation 18

The required information to access the current entry of the FIFO is given in the following registers:

- *The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.*

  The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

$$RF\_DFO[X] = FR\_RFSDOR[X] + (FR\_RFDSR[X][ENTRY\_SIZE] \times 2) \times i - FR\_RFSIDX[X])$$

Equation 19

Note

The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to FIFO Update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

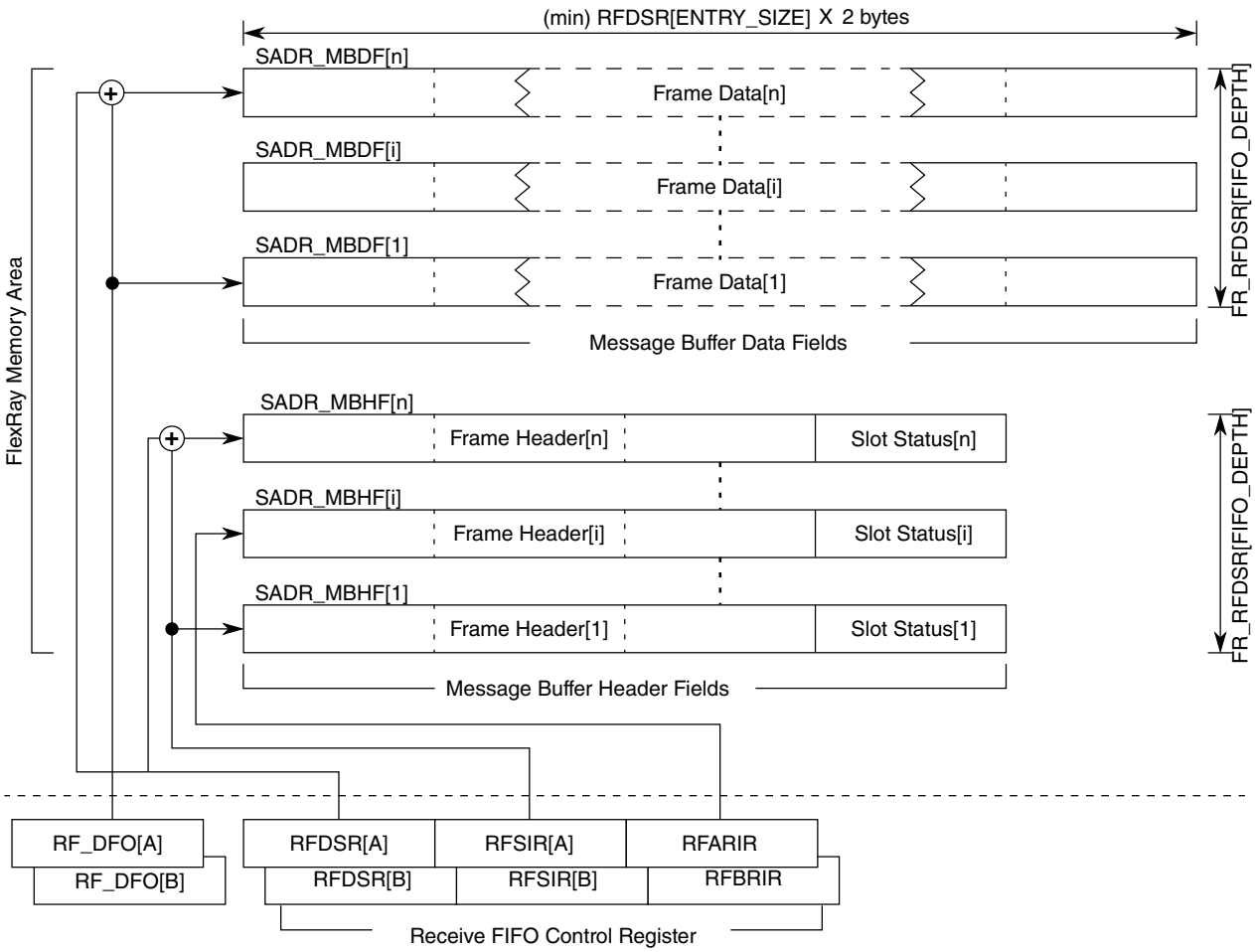


Figure 60-5. Receive FIFO Structure

Note

The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 19 on page 3041. They are not stored in a register.

60.8.3.4 Message Buffer Configuration and Control Data

This section describes the configuration and control data for each message buffer type.

60.8.3.4.1 Individual Message Buffer Configuration Data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

60.8.3.4.1.1 Common Configuration Data

The set of common configuration data for individual message buffers is located in the following registers.

- *Message Buffer Data Size Register (FR_MBDSR)*

  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.

- *Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)*

  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, see Individual Message Buffer Segments.

60.8.3.4.1.2 Specific Configuration Data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- *Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)*

  The MTD bit configures the message buffer type.

- *Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)*

  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.

- *Message Buffer Frame ID Registers (FR_MBFIDRn)*

  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.

- *Message Buffer Index Registers (FR_MBIDXRn)This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.*

60.8.3.5 Individual Message Buffer Control Data

During normal operation, each individual message buffer can be controlled by the control and trigger bits CMT, LCKT, EDT, and MBIE in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn).

60.8.3.6 Receive Shadow Buffer Configuration Data

Before frame reception into the individual message buffers can be performed, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

60.8.3.7 Receive FIFO Control and Configuration Data

This section describes the configuration and control data for the two receive FIFOs.

60.8.3.7.1 Receive FIFO Configuration Data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- *Receive FIFO Periodic Timer Register (FR_RFPTR)*

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- *Receive FIFO Watermark and Selection Register (FR_RFWMSR)*
- *Receive FIFO Start Index Register (FR_RFSIR)*
- *Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *Receive FIFO Depth and Size Register (FR_RFDSR)*
- *Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)*
- *Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)*
- *Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)*

- *Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*

60.8.3.7.2 Receive FIFO Control Data

The application can access the FIFOs at any time using the control bits in the following registers:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

60.8.3.7.3 Receive FIFO Status Data

The current status of the receive fifo is provided in the following register:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO A Read Index Register (FR_RFARIR)*
- *Receive FIFO B Read Index Register (FR_RFBRIR)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

60.8.4 FlexRay Memory Area Layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

60.8.4.1 FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

In the figure given below, shows an example layout for the FIFO address mode FR_MCR[FAM]=0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is maximum 64 KB.
- The FlexRay memory area starts at a 16 byte boundary

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

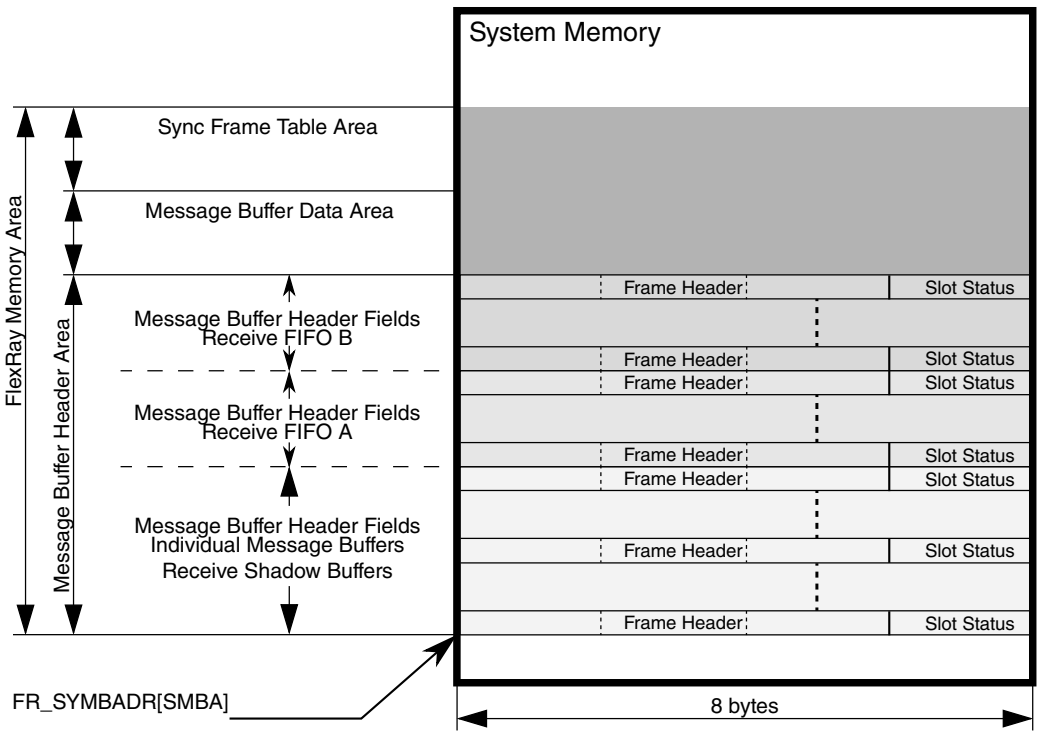


Figure 60-6. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

60.8.4.2 FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

The following figure shows an example layout for the FIFO address mode FR_MCR[FAM]=1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

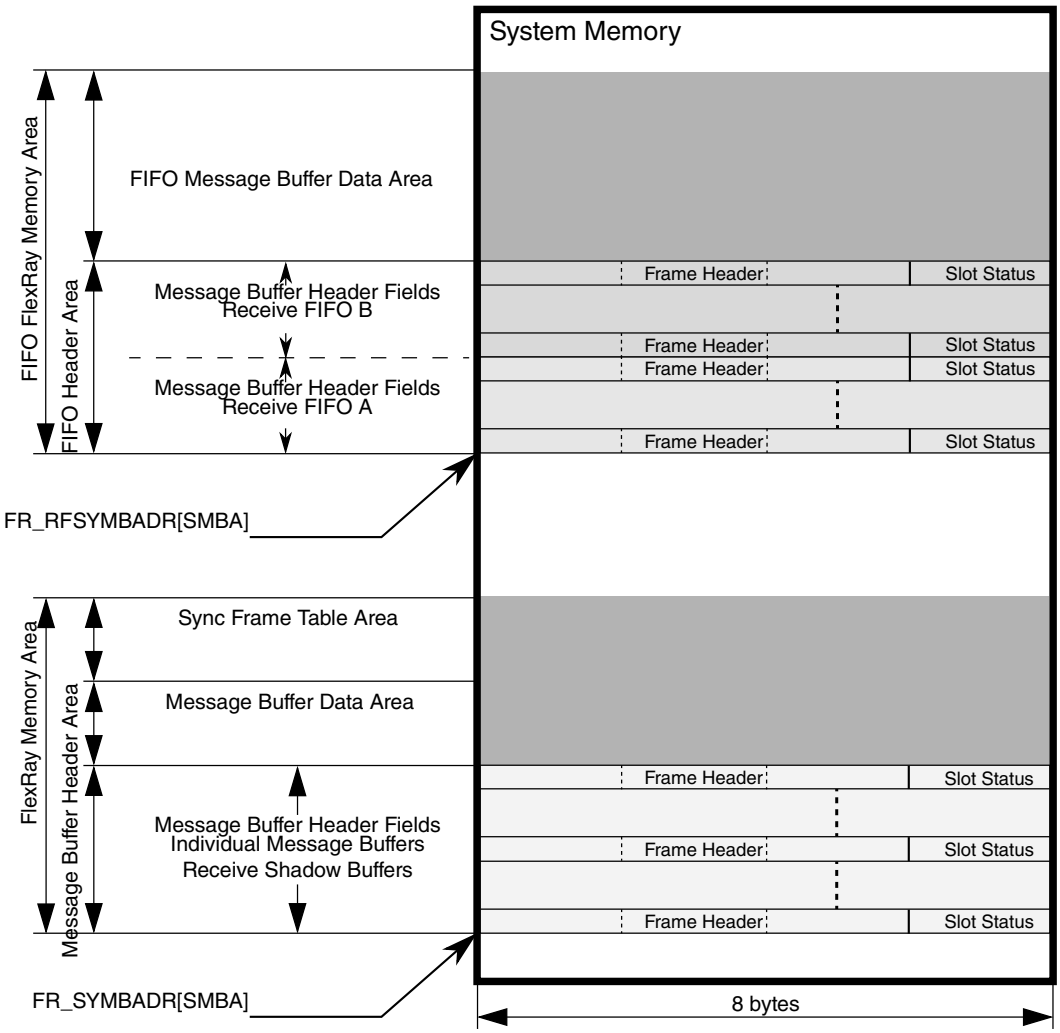


Figure 60-7. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

60.8.4.3 Message Buffer Header Area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rules apply to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for *individual message buffers* and *receive shadow buffers* must fulfill Equation 20 on page 3047.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 20

2. The start byte address SADR_MBHF of each message buffer header field for the *FIFO* must fulfill Equation 20 on page 3047.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 21

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 22

3. The message buffer header fields for each FIFO have to be a contiguous area.

60.8.4.4 Message Buffer Header Area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill Equation 23 on page 3048.

$$SADR\_MBHF = (i \times 8) + FR\_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 23

60.8.4.5 FIFO Message Buffer Header Area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill the following equation:Equation 24 on page 3048.

$$SADR\_MBHF = (i\gamma 8) + FR\_RFSYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 24

2. The message buffer header fields for each FIFO have to be a contiguous area.

60.8.4.6 Message Buffer Data Area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

60.8.4.7 Sync Frame Table Area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to Sync Frame ID and Sync Frame Deviation Tables for the description of the sync frame table area.

60.8.5 Physical Message Buffer Description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer, the message buffer header field and the message buffer data field.

60.8.5.1 Message Buffer Protection and Data Consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

60.8.5.2 Message Buffer Header Field Description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in Message Buffer Header Field. Each message buffer header field consists of two sections: the frame header section and the slot status section.

60.8.5.2.1 Frame Header Description

Frame header content, access, and checks are discussed in this section.

60.8.5.2.1.1 Frame Header Content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in the figure below. A detailed description is given in Table 60-23.

Table 60-19. Frame Header Structure (Receive Message Buffer and Receive FIFO)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2	0	0	CYCCNT						0	PLDLEN						
0x4	0	0	0	0	0	HDCRC										

The structure of the frame header in the message buffer header field for transmit message buffers is given in the following figure. A detailed description is given in Table 60-24. The checks that will be performed are described in Frame Header Checks.

Table 60-20. Frame Header Structure (Transmit Message Buffer)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID [1]										
0x2			CYCCNT							PLDLEN [2]						
0x4						HDCRC										

= not used

1. checked
2. checked if not static

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in the figure below.

Table 60-21. Frame Header Structure (Transmit Message Buffer for Key Slot)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2			CYCCNT							PLDLEN						
0x4						HDCRC										

= not used

60.8.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in the following table. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 60-22. Frame Header Write Access Constraints (Transmit Message Buffer)

Field	Static Segment	Dynamic Segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	MB_LCK

60.8.5.2.1.3 Frame Header Checks

As shown in Table 60-20 and Table 60-21 not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDRn). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag FID_EF in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDRn).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 60-23. Frame Header Field Descriptions (Receive Message Buffer and Receive FFO)

Field	Description
R	**Reserved Bit** — This is the value of the *Reserved bit* of the received frame stored in the message buffer
PPI	**Payload Preamble Indicator** — This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	**Null Frame Indicator** — This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	**Sync Frame Indicator** — This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	**Startup Frame Indicator** — This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	**Frame ID** — This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	**Cycle Count** — This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	**Payload Length** — This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	**Header CRC** — This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 60-24. Frame Header Field Descriptions (Transmit Message Buffer)

Field	Description
R	**Reserved Bit** — This bit is not used, the value of the *Reserved bit* is generated internally according to *FlexRay Communications System Protocol Specification, Version*2.1 Rev A.
PPI	**Payload Preamble Indicator** — This bit provides the value of the *Payload Preamble Indicator* for the frame transmitted from the message buffer.
NUF	**Null Frame Indicator** — This bit is not used, the value of the *Null Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SYF	**Sync Frame Indicator** — This bit is not used, the value of the *Sync Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SUF	**Startup Frame Indicator** — This bit is not used, the value of the *Startup Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
FID	**Frame ID** — This field is checked as described in Frame Header Checks.
CYCCNT	**Cycle Count** — This field is not used, the value of the transmitted *Cycle Count* field is taken from the internal communication cycle counter.
PLDLEN	**Payload Length** — This field is checked and used as described in Frame Header Checks.

Table continues on the next page...

Table 60-24. Frame Header Field Descriptions (Transmit Message Buffer) (continued)

Field	Description
HDCRC	**Header CRC** — This field provides the value of the *Header CRC* field for the frame transmitted from the message buffer.

60.8.5.2.2 Slot Status Description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

60.8.5.2.2.1 Receive Message Buffer and Receive FIFO Slot Status Description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given in the next table.

Table 60-25. Receive Message Buffer Slot Status Content

Receive Message Buffer Type	Slot Status Content
Individual Receive Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 60-26
Individual Receive Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 60-27
Individual Receive Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 60-28
Receive FIFO Channel A Message Buffer	see Table 60-27
Receive FIFO Channel B Message Buffer	see Table 60-28

The meaning of the bits in the slot status structure is explained in the Receive Message Buffer Slot Status Field Description table below.

Table 60-26. Receive Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-27. Receive Message Buffer Slot Status Structure (ChA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table continues on the next page...

Table 60-27. Receive Message Buffer Slot Status Structure (ChA) (continued)

R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-28. Receive Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-29. Receive Message Buffer Slot Status Field Description

Field	Description
	Common Message Buffer Status Bits
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	**Channel first valid received** — This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 first valid frame received on channel A, or no valid frame received at all

Table continues on the next page...

Table 60-29. Receive Message Buffer Slot Status Field Description (continued)

Field	Description
	1 first valid frame received on channel B
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

60.8.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given in the table below.

Table 60-30. Transmit Message Buffer Slot Status Content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 60-31

Table continues on the next page...

Table 60-30. Transmit Message Buffer Slot Status Content (continued)

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 60-32
Individual Transmit Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 60-33

The meaning of the bits in the slot status structure is described in the Transmit Message Buffer Slot Status Structure Field Descriptions table.

Table 60-31. Transmit Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-32. Transmit Message Buffer Slot Status Structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-33. Transmit Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 60-34. Transmit Message Buffer Slot Status Structure Field Descriptions

Field	Description
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table continues on the next page...

Table 60-34. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	**Transmission Conflict on Channel B** — protocol related variable: *vSS!TxConflict* channel B 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0

Table continues on the next page...

Table 60-34. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
	1 *vSS!BViolation* = 1
TCA	**Transmission Conflict on Channel A** — protocol related variable: *vSS!TxConflict* channel A 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

60.8.5.3 Message Buffer Data Field Description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in the table below. The structure of the message buffer data field is given in the next figure.

Table 60-35. Message Buffer Data Field Minimum Length

physical message buffer assigned to	minimum length defined by
Individual Message Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Receive Shadow Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Individual Message Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive Shadow Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note

The CC will not access any locations outside the message buffer data field boundaries given in the table above.

Table 60-36. Message Buffer Data Field Structure

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	DATA0 / MID0 / NMV0								DATA1 / MID1 / NMV1							
0x2	DATA2 / NMV2								DATA3 / NMV3							
…	…								…							
0xN-2	DATA N-2								DATA N-1							

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

60.8.5.3.1 Message Buffer Data Field Read Access

For transmit message buffers, the CC will not modify the content of the Message Buffer Data Field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXRn). While the message buffer is locked, the CC will not update the Message Buffer Data Field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

60.8.5.3.2 Message Buffer Data Field Write Access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in the Frame Data Write Access Constraints table below.

Table 60-37. Frame Data Write Access Constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 60-38. Frame Data Field Descriptions

Field	Description
DATA 0, DATA 1, … DATA N-1	**Message Data** — Provides the message data received or to be transmitted.For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	**Message Identifier** — If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.

Table continues on the next page...

Table 60-38. Frame Data Field Descriptions (continued)

Field	Description
NMV 0, NMV 1, … NMV 11	**Network Management Vector** — If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

60.8.6 Individual Message Buffer Functional Description

The CC provides these basic types of individual message buffers:

1. Transmit Message Buffers
2. Receive Message Buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in Individual Message Buffer Configuration Data.

60.8.6.1 Individual Message Buffer Configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

60.8.6.1.1 Common Configuration Data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. These data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

60.8.6.1.2 Specific Configuration Data

The second part of the message buffer configuration data is specific for each message buffer.

These data can be changed only when either

- *the protocol is in the POC:config state or*
- *the message buffer is disabled, i.e. FR_MBCCSRn[EDS] = 0*

The individual message buffer type is defined by the MTD and MBT bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) as given in the following table.

Table 60-39. Individual Message Buffer Types

FR_MBCCSRn		Individual Message Buffer Description
MTD	MBT	
0	0	Receive Message Buffer
0	1	Reserved
1	0	Transmit Message Buffer
1	1	Reserved

The message buffer specific configuration data are

1. MTD bits in Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
2. all fields and bits in Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)

3. all fields and bits in Message Buffer Frame ID Registers (FR_MBFIDRn)
4. all fields and bits in Message Buffer Index Registers (FR_MBIDXRn)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions in the Transmit Message Buffers section and Receive Message Buffers section.

60.8.6.2 Transmit Message Buffers

The section provides a detailed description of the functionality of single buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number n is configured to be a transmit message buffer by the following settings:

- FR__ MBCCSRn[MBT] = 0(singlebufferedmessagebuffer)
- FR_ MBCCSRn[MDT] = 1(transmitmessagebuffer)

60.8.6.2.1 Access Regions

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented, which is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in the figure below. A description of the regions is given in Table 60-41. If an region is active as indicated in Table 60-42, the access scheme given for that region applies to the message buffer.

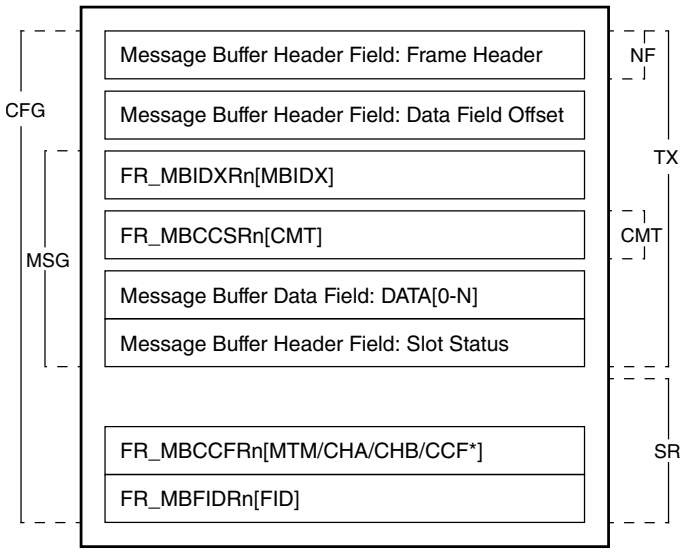


Figure 60-8. Transmit Message Buffer Access Regions

Table 60-40. Transmit Message Buffer Access Regions Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration
MSG	read/write	-	Message Data and Slot Status Access
NF	-	read-only	Message Header Access for Null Frame Transmission
TX	-	read/write	Message Transmission and Slot Status Update
CM	-	read-only	Message Buffer Validation
SR	-	read-only	Message Buffer Search

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT], and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in the next figure. A description of the states is given in Table 60-41, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

60.8.6.2.2 Message Buffer States

This section describes the transmit message buffer states and provides a state diagram.

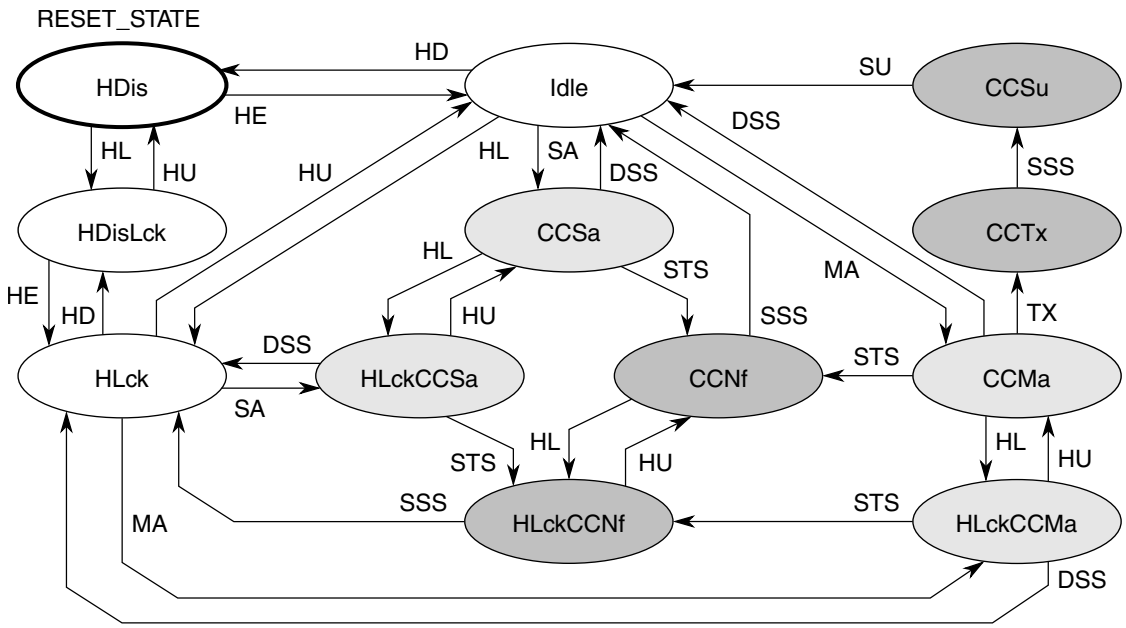


Figure 60-9. Transmit Message Buffer States

Table 60-41. Transmit Message Buffer State Description (Sheet 1 of 2)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	CM,SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	SR	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCSa	1	0	–	–	**Slot Assigned** - Message buffer assigned to next static slot.Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	–	**Locked and Slot Assigned** - Applications access to data, control, and status.Message buffer assigned to next static slot
CCNf	1	0	–	NF	**Null Frame Transmission** Header is used for null frame transmission.

Table continues on the next page...

Table 60-41. Transmit Message Buffer State Description (Sheet 1 of 2) (continued)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
HLckCCNf	1	1	MSG	NF	**Locked and Null Frame Transmission** - Applications access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	–	CM	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
HLckCCMa	1	1	MSG	–	**Locked and Message Available** - Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	–	TX	**Message Transmission** - Message buffer data transmit. Payload data from buffer transmitted
CCSu	1	0	–	TX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

60.8.6.2.3 Message Buffer Transitions

Application transitions, module transitions, and transition priorities are discussed in this section.

60.8.6.2.3.1 Application Transitions

The application transitions can be triggered by the application using the commands described in the table below. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

60.8.6.2.3.1.1 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

60.8.6.2.3.1.2 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 60-42. Transmit Message Buffer Application Transitions

Transition	Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]:= 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

60.8.6.2.3.2 Module Transitions

The module transitions that can be triggered by the CC are described in the following table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 60-43. Transmit Message Buffer Module Transitions

Transition	Condition	Description
SA	slot match andstatic slot	**Slot Assigned** - Message buffer is assigned to next static slot.
MA	slot match andCycleCounter match	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
TX	slot start and FR_MBCCSRn[CMT] = 1	**Transmission Slot Start** - Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	status updated	**Status Updated** - Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	static slot start	**Static Slot Start** - Start of static slot.
DSS	dynamic slot start orsymbol window start orNIT start	**Dynamic Slot or Segment Start.** - Start of dynamic slot or symbol window or NIT.
SSS	slot start orsymbol window start orNIT start	**Slot or Segment Start** - Start of static slot or dynamic slot or symbol window or NIT.

60.8.6.2.3.3 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of the next table, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and subsequently the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions is given in the second part of the table below.

Table 60-44. Transmit Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle, HLck	SA > HD	Slot Assigned > Message Buffer Disable
	MA > HD	Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS	Transmission Slot Start > Static Slot Start
	TX > DSS	Transmission Slot Start > Dynamic Slot Start

60.8.6.2.4 Transmit Message Setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit CMT in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). The physical access to the message buffer data field is described in Individual Message Buffers.

As indicated by Table 60-34, the application shall write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in Table 60-34. The state change is indicated through the FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa the FR_MBCCSRn[DVAL] flag is negated.

60.8.6.2.5 Message Transmission

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. the message buffer is in the message available state CCMa
2. the message data are still valid, i.e. FR_MBCCSRn[CMT] = 1

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in the "Message Transmission Timing" figure below. In this example, the message buffer with message buffer number n is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid, i.e. FR_MBCCSRn[CMT] = 1.

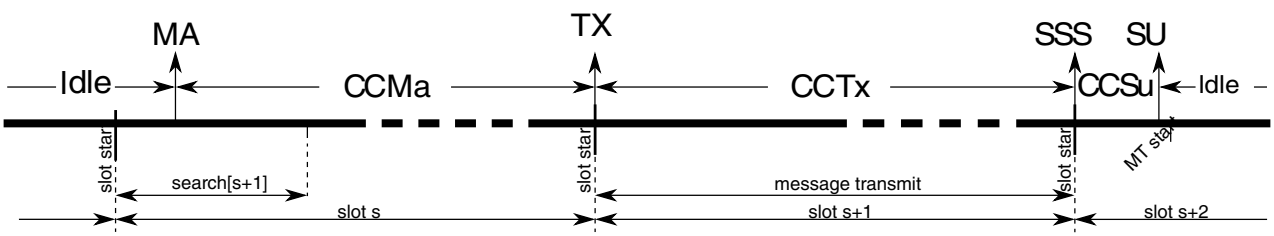


Figure 60-10. Message Transmission Timing

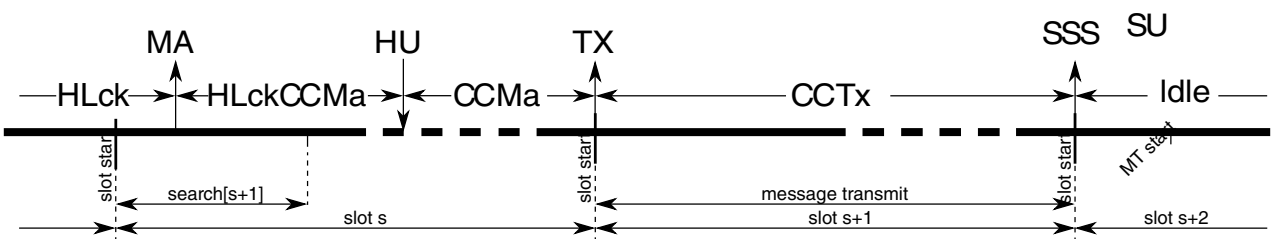


Figure 60-11. Message Transmission from HLck state with unlock

The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)

3. the value of the PLDLEN field in the message buffer header field, as described in Frame Header Description

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN <= MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

60.8.6.2.6 Null Frame Transmission

A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDRn[FID] set to S and FR_MBCCFRn[CHA] set to 1. A Null Frame is transmitted in the static slot S on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDRn[FID] = s and FR_MBCCFRn[CHA] = 1 are either not committed, i.e FR_MBCCSRn[CMT] = 0, or locked by the application, i.e. FR_MBCCSRn[LCKS] = 1, or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in the figure below.

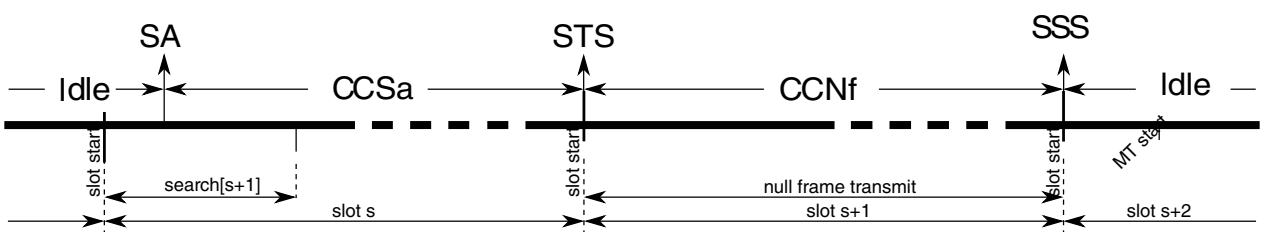


Figure 60-12. Null Frame Transmission from Idle state

A message buffer timing and state change diagram for null frame transmission from HLck state is given in the following figure.

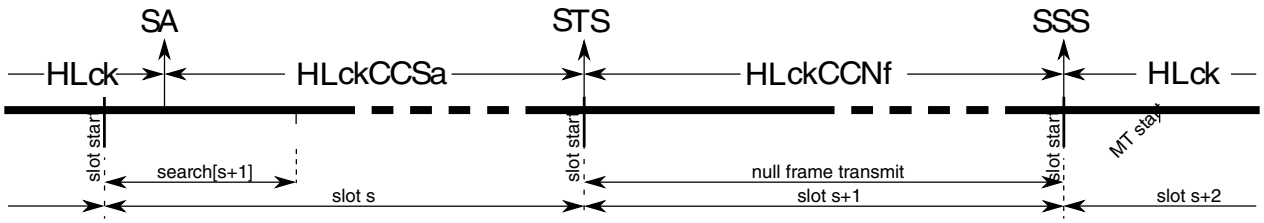


Figure 60-13. Null Frame Transmission from HLck state

If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the next figure.

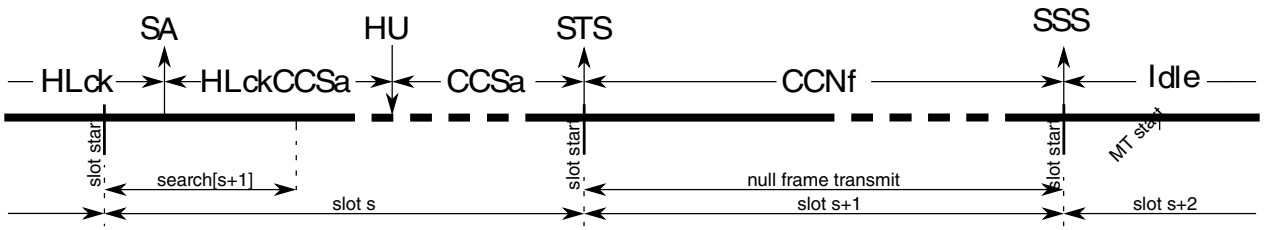


Figure 60-14. Null Frame Transmission from HLck state with unlock

Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the figure below.

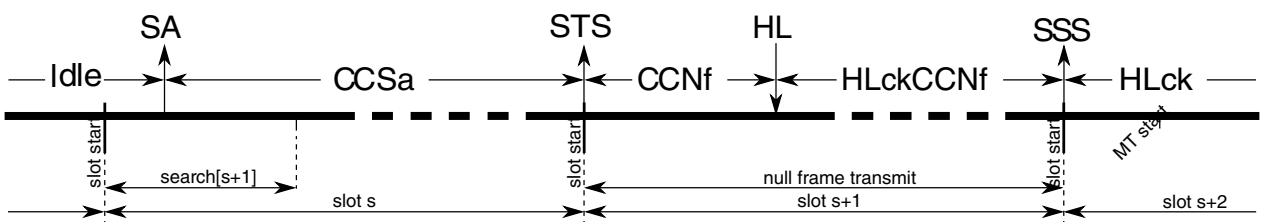


Figure 60-15. Null Frame Transmission from Idle state with locking

60.8.6.2.7 Message Buffer Status Update

After the end of each slot, the PE generates the slot status vector. Depending on the this status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

60.8.6.2.7.1 Message Buffer Status Update after Complete Message Transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer were send to FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSRn[MBIF] to indicate the successful message transmission.

Depending on the transmission mode flag FR_MBCCFRn[MTM], the CC changes the commit flag FR_MBCCSRn[CMT] and the valid flag FR_MBCCSRn[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in the *event transmission mode*. In this case, each committed message is transmitted only once. The commit flag FR_MBCCSRn[CMT] is cleared with the SU transition. If the FR_MBCCFRn[MTM] flag is asserted, the message buffer is in the *state transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSRn[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSRn[DVAL] to indicate that the message will be transmitted again.

60.8.6.2.7.2 Message Buffer Status Update after Incomplete Message Transmission

The term incomplete message transmission refers to the fact that not all payload data that should be transmitted were send to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

In any of these two aforementioned conditions occur, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those errors occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

60.8.6.2.7.3 Message Buffer Status Update after Null Frame Transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

60.8.6.3 Receive Message Buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used it is required to configure the related receive shadow buffer as described in Receive Shadow Buffers.

A receive message buffer is used to receive a message from the FlexRay Bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application.

1. message data received
2. information about the reception process
3. status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration settings

$$FR\_MBCCSRn[MTD] = 0 (receive message buffer)$$

Equation 25

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented that is used to control the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in the figure below. A description of the regions is given in the following table. If a region is active as indicated in the "Receive Message Buffer States and Access (Sheet 2 of 2)" table below, the access scheme given for that region applies to the message buffer.

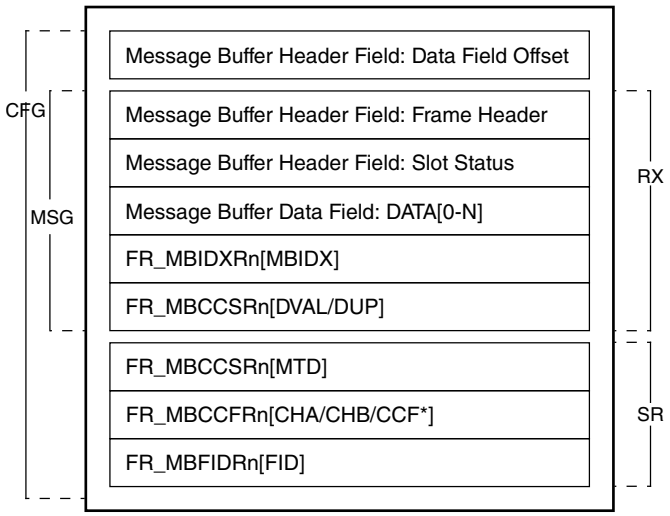


Figure 60-16. Receive Message Buffer Access Regions

Table 60-45. Receive Message Buffer Access Region Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration, Message Data and Status Access
MSG	read/write	-	Message Data, Header, and Status Access
RX	-	write-only	Message Reception and Status Update
SR	-	read-only	Message Buffer Search Data

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT] and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in the figure below. A description of the message buffer states is given in Table 60-41, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

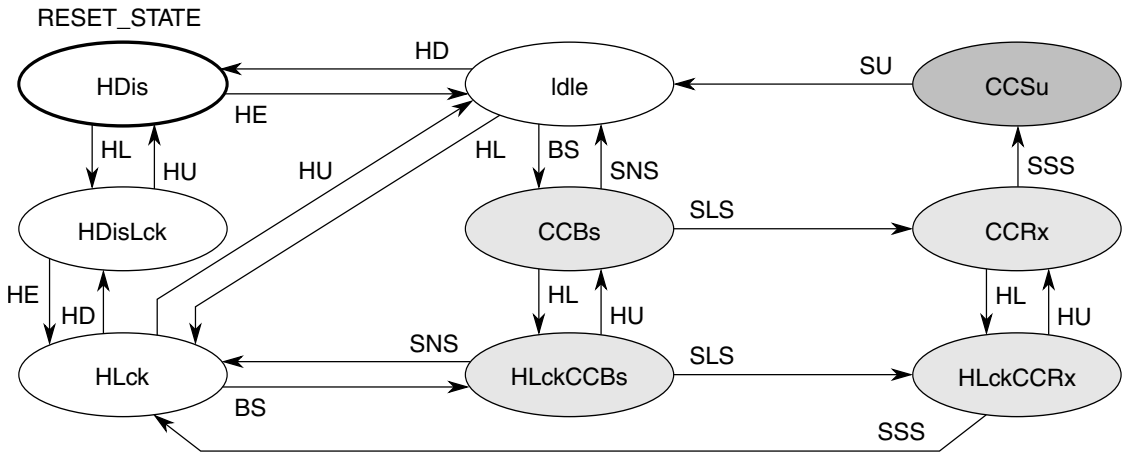


Figure 60-17. Receive Message Buffer States

Table 60-46. Receive Message Buffer States and Access (Sheet 2 of 2)

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	–	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCBs	1	0	–	–	**Buffer Subscribed** - Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	–	**Locked and Buffer Subscribed** - Applications access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	–	–	**Message Receive** - Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	–	**Locked and Message Receive** - Applications access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	–	RX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

60.8.6.3.1 Message Buffer Transitions

Application transitions, message buffer enable and disable, message buffer lock and unlock, module transitions, and transition priorities are discussed in this section.

60.8.6.3.1.1 Application Transitions

The application transitions that can be triggered by the application using the commands described in Table 60-42. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

60.8.6.3.1.2 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCBs, HLckCCBs, or CCRx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

60.8.6.3.1.3 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL while the message buffer is in the state CCRx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 60-47. Receive Message Buffer Application Transitions

Transition	Host Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]: = 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

60.8.6.3.1.4 Module Transitions

The module transitions that can be triggered by the CC are described in the next table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 60-48. Receive Message Buffer Module Transitions

Transition	Condition	Description
BS	slot match and CycleCounter match	**Buffer Subscribed** - The message buffer filter matches next slot and cycle.
SLS	slot start	**Slot Start** - Start of either Static Slot or Dynamic Slot.
SNS	symbol window start or NIT start	**Symbol Window or NIT Start** - Start of either Symbol Window or NIT.
SSS	slot start or symbol window start or NIT start	**Slot or Segment Start** - Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	status updated	**Status Updated** - Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

60.8.6.3.1.5 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the table below, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU and can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the buffer subscribed state CCBs and the module triggers the slot start transition SLS at the same time as the application locks the message buffer by the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state HLckCCRx.

Table 60-49. Receive Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle	BS > HD	Buffer Subscribed > Message Buffer Disable
HLck	BS > HD	Buffer Subscribed > Message Buffer Disable
CCRx	SSS > HL	Slot or Segment Start > Message Buffer Lock

60.8.6.3.2 Message Reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to the either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffers assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot the module trigger the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data are written into the shadow buffers. For details on receive shadow buffers, see Receive Shadow Buffers Concept. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

60.8.6.3.3 Message Buffer Update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer was in the locked state HLckCCRx, no update will be performed. The received data are lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state it is now in the CCSu state. After the evaluation of the slot status provided by the PE the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to. This is described in Table 60-52.

Table 60-50. Receive Message Buffer Update (Continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
1	1	**Valid non-null frame received.** - Message Buffer Data Field updated. - Frame Header Field updated. - Slot Status Field updated. - DUP:= 1 - DVAL:= 1 - MBIF:= 1
1	0	**Valid null frame received.**

Table continues on the next page...

Table 60-50. Receive Message Buffer Update (Continued) (continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
		- Message Buffer Data Field *not* updated. - Frame Header Field *not* updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed - MBIF:= 1
0	x	**No valid frame received.** - Message Buffer Data Field not updated. - Frame Header Field not updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed. - MBIF:= 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: *vSS!ValidFrame* = 0 and *vSS!SyntaxError* = 0 and *vSS! ContentError* = 0 and *vSS!BViolation* = 0.

Note

If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > *n* will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in the following figure.

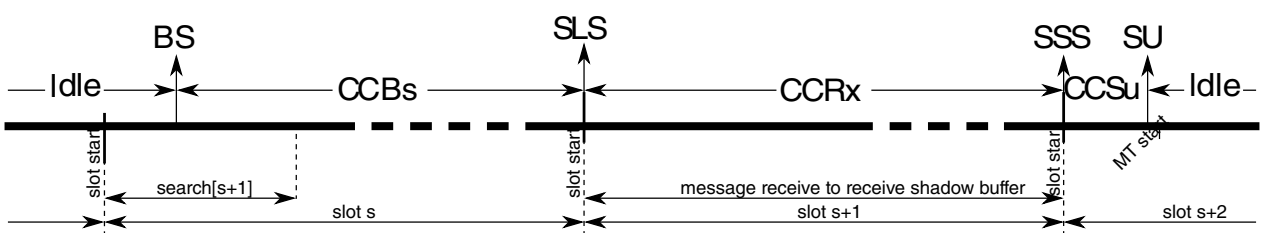


Figure 60-18. Message Reception Timing

The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following two items:

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. the number of bytes received over the FlexRay bus

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

60.8.6.3.4 Received Message Access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in Individual Message Buffers.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

60.8.6.3.5 Receive Shadow Buffers Concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in Receive Shadow Buffers.

The receive shadow buffers temporarily store the received frame header and message data. After the slot boundary the slot status information is generated. If the slot status information indicates the reception of the valid non-null frame (see Table 60-50), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXRn) with the content of the corresponding internal shadow buffer index register. In all other cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXRn). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXRn).

60.8.7 Individual Message Buffer Search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel if a slot s in a communication cycle c is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm which is invoked at the following protocol related events:

1. NIT start
2. slot start in the static segment
3. minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *n* searches for message buffers assigned or subscribed to slot *n+1*.

In general, the message buffer search for the next slot *n* considers only message buffers which are

1. enabled, i.e. FR_MBCCSRn[EDS] = 1, and
2. matches the next slot *n*, i.e. FR_MBFIDRn[FID] = *n*, and

On top of that, for the static segment only those message buffers are considered, that match the condition of at least one row of the "Message Buffer Search Priority (static segment)" table shown below. For the dynamic segment only those message buffers are considered, that match the condition of at least one row of the next "Message Buffer Search Priority (dynamic segment)" table. These message buffers are called *matching* message buffers.

For each enabled channel the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers the message buffers with highest priority according to the "Message Buffer Search Priority (static segment)" table for the static segment and according to the "Message Buffer Search Priority (dynamic segment)" table for the dynamic segment are selected.

Table 60-51. Message Buffer Search Priority (static segment)

Priority	MTD	LCKS	CMT	CCFM[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	1	-	0	1	transmit buffer, matches cycle count, not committed	SA
	1	1	-	1	transmit buffer, matches cycle count, locked	SA
2	1	-	-	-	transmit buffer	SA
3	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 4	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

Table 60-52. Message Buffer Search Priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 2	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffer which have the highest priority must have a consistent channel assignment as specified in Message Buffer Channel Assignment Consistency.

Depending on the message buffer channel assignment the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in Individual Message Buffers.

60.8.7.1 Message Buffer Cycle Counter Filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn). This filter determines a set of communication cycles in which the

message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled, i.e. CCFE = 0, this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

$$MBCCFRn[CCFE] = 0$$

$$CYCCNT \,\&\, MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \,\&\, MBCCFRn[CCFMSK]$$

60.8.7.2 Message Buffer Channel Assignment Consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFRn[CHA] = 1 and transmits or receives on channel B if FR_MBCCFRn[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined, if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in the figure below.

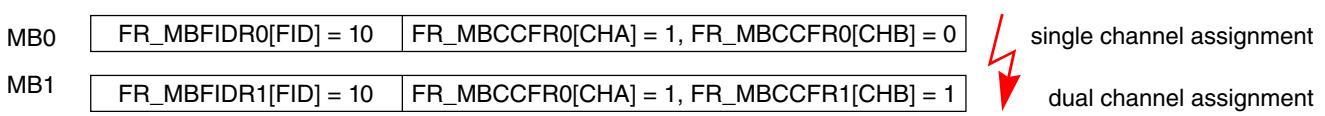


Figure 60-19. Inconsistent Channel Assignment

60.8.7.3 Node Related Slot Multiplexing

The term *Node Related Slot Multiplexing* applies to the dynamic segment only and refers to the functionality if there are transmit as well as receive message buffers are configured for the same slot.

According to Table 60-52 the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

60.8.7.4 Message Buffer Search Error

There are two kinds of errors which may occur during message buffer search[1].

60.8.7.4.1 Message Buffer Search Start while Running

If the message buffer search is running in slot *n-1* and the next message buffer search start event appears due to the start of slot *n*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *n*. Additionally, the search engine will not be started in slot *n*, and consequently no individual message buffer is identified for transmission or reception in slot *n+1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see Number of Usable Message Buffers.

60.8.7.4.2 Illegal Message Buffer Index Found

If the message buffer search has finished the message buffer search in slot *n-1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) is set and no individual message buffer is identified for transmission or reception in slot *n*. The legal message buffer index values for the individual and receive shadow buffers are specified in the Receive Shadow Buffer Index Register section (FR_RSBIR) and the Message Buffer Index Registers sections (FR_MBIDXRn).

1. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

60.8.8 Individual Message Buffer Reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data can not be reconfigured when the protocol is out of the *POC:config* state.

60.8.8.1 Reconfiguration Schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

60.8.8.1.1 Basic Type Not Changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSRn[MTD] are not changed. This type of reconfiguration is denoted by RC1 in Figure 60-20. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

60.8.8.1.2 Buffer Type Not Changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in the figure below. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

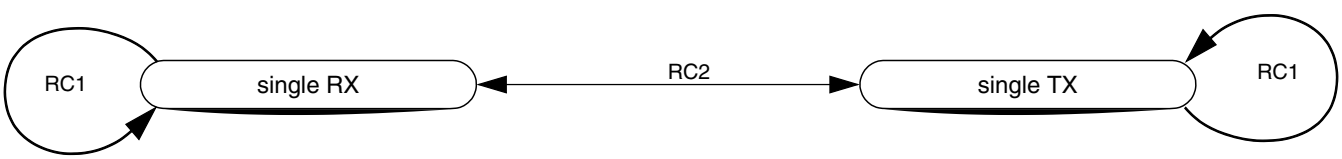


Figure 60-20. Message Buffer Reconfiguration Scheme

60.8.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

60.8.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. One FIFO is assigned to channel A, the other FIFO is assigned to channel B. Both FIFOs work completely independent from each other.

The message buffer structure of each FIFO is described in Receive FIFO. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- *The FIFO system memory base address*
- *The index of the first FIFO entry given by Receive FIFO Start Index Register (FR_RFSIR)*
- *The data field offset of the data field belonging to the first FIFO entry given by Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *The number of FIFO entries and the length of each FIFO entry as given by Receive FIFO Depth and Size Register (FR_RFDSR)*

60.8.9.2 FIFO Configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO address mode bit FAM in the Module Configuration Register (FR_MCR).

60.8.9.2.1 Single System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is System Memory Base Address Register (FR_SYMBADR).

60.8.9.2.2 Dual System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data are given in Receive FIFO Control and Configuration Data. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details see FlexRay Memory Area Layout.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

- Configure the FIFO update and address modes in Module Configuration Register (FR_MCR)
- Configure the FIFO system memory base address
- Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO
- Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth
- Configure the FIFO Filters

60.8.9.3 FIFO Periodic Timer

The FIFO periodic timer is used to generate an FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached, but the FIFO is not empty. This can be used to prevent frames from get stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd*, greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

60.8.9.4 FIFO Reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO, if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case the FIFO filter path shown in Figure 60-21 is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented. If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

60.8.9.5 FIFO Almost-Full Interrupt Generation

If the fifo fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM, i.e. FLA>WM_A (FLB>WM_B), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

If the periodic timer expires, and FIFOA (FIFOB) is not empty, i.e. FLA>0 (FLB>0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

60.8.9.6 FIFO Overflow Error Generation

If the FIFOA (FIFOB) is full, i.e. FLA=$FIFO\_DEPTH_A$ (FLB=$FIFO\_DEPTH_B$) and the conditions for a FIFO reception as described in FIFO Reception are fulfilled, then the fifo overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

60.8.9.7 FIFO Message Access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the

(Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to Equation 1-125.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, than the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to Equation 1-125. The application can access the message data as described in Receive FIFO. When the application has read the message buffer data and status information, it can update the FIFO as described in FIFO Update.

60.8.9.8 FIFO Update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

60.8.9.8.1 FIFO Interrupt Flag Update

The FIFO Interrupt Flag Update mode is configured, when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode FIFOA (FIFOB) will be updated by 1 entry, when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1, if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

60.8.9.9 FIFO Filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides three sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in the figure below.

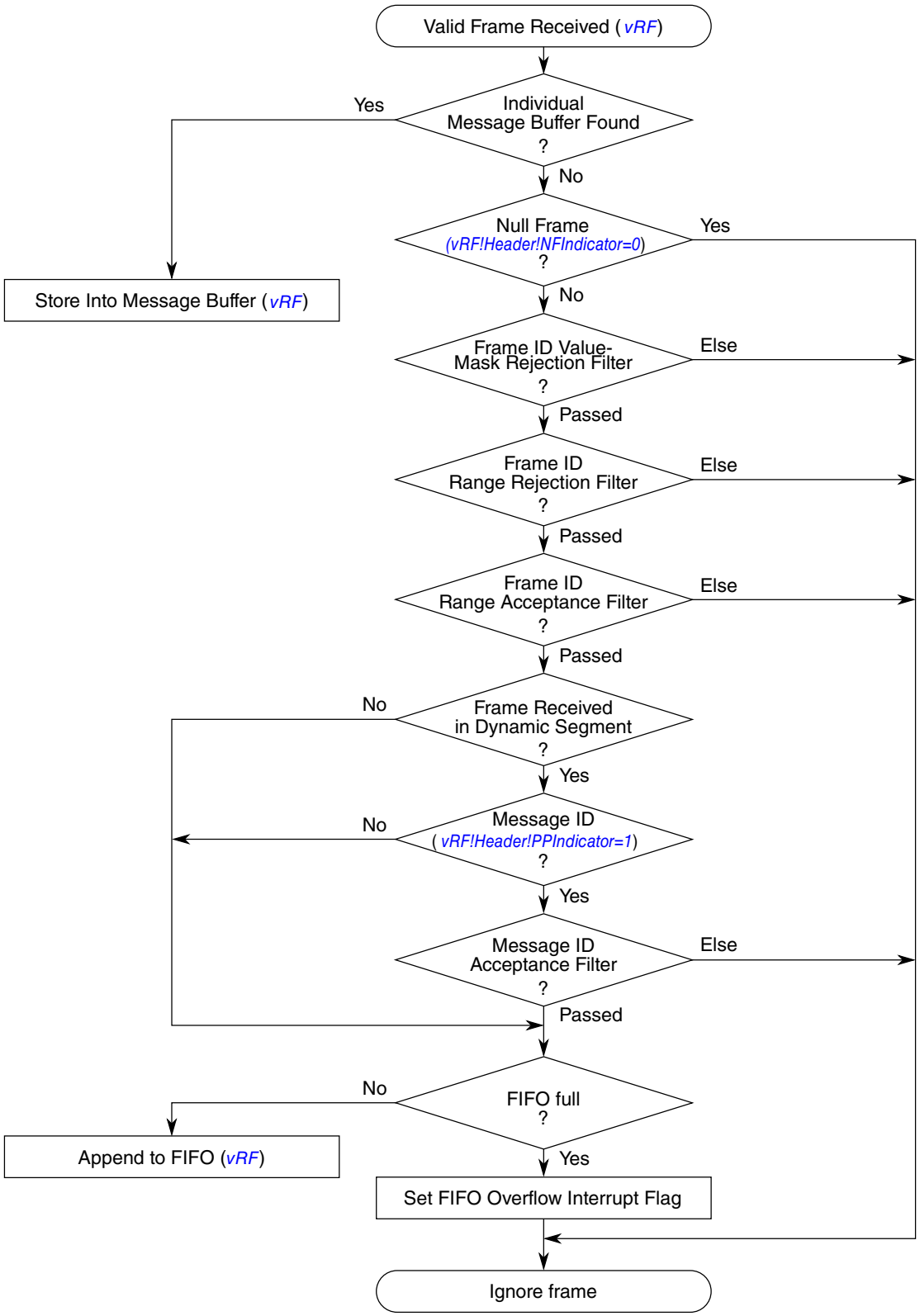


Figure 60-21. Received Frame FIFO Filter Path

A received frame passes the FIFO filtering if it has passed all three type of filter.

60.8.9.9.1 RX FIFO Frame ID Value-Mask Rejection Filter

The frame ID value-mask rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter, i.e. is not rejected.

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-Mask Rejection Filter if Equation 26 on page 3091 is fulfilled.

$$FID \,\&\, FR\_RFFIDRFMR[FIDRFMSK] \neq FR\_RFFIDRFVR[FIDRFVAL] \,\&\, FR\_RFFIDRFMR[FIDRFMSK]$$

Equation 26

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to pass all frames by the following settings.

$$FR\_RFFIDRFVR[FIDRFVAL] = 0x000 \text{ and } FR\_RFFIDRFMR[FIDRFMSK] = 0x7FF$$

Equation 27

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to reject all frames by the following settings.

$$FR\_RFFIDRFMR[FIDRFMSK] = 0x000$$

Equation 28

Using these settings, Equation 26 on page 3091 can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

60.8.9.9.2 RX FIFO Frame ID Range Rejection Filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range rejection filters if either no rejection filter is enabled, or, for all of the enabled RX FIFO Frame ID Range rejection filters, i.e. FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1, Equation 29 on page 3092 is fulfilled.

$$FID < FR\_RFRFCFR_{SEL}[SID_{IBD=0}]) or (FR\_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Equation 29

Consequently, all frames with a frame ID that fulfills Equation 30 on page 3092 for at least one of the enabled rejection filters will be rejected and thus not pass.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 30

60.8.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, for at least one of the enabled RX FIFO Frame ID Range acceptance filters, i.e. FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1, Equation 31 on page 3092 is fulfilled.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 31

60.8.9.9.4 RX FIFO Message ID Acceptance Filter

The RX FIFO Message ID Acceptance Filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance Filter if Equation 32 on page 3092 is fulfilled.

$$MID \,\&\, FR\_RFMIDAFMR[MIDAFMSK] = FR\_RFMIDAFMR[MIDAFVAL] \,\&\, FR\_RFMIDAFMR[MIDAFMSK]$$

Equation 32

The RX FIFO Message ID Acceptance Filter can be configured to accept all frames by setting

$$FR\_RFMIDAFMR[MIDAFMSK] := 0x000$$

Equation 33

Using these settings, Equation 32 on page 3092 is always fulfilled and all frames will pass.

60.8.10 Channel Device Modes

This section describes the two FlexRay channel device modes that are supported by the CC.

60.8.10.1 Dual Channel Device Mode

In the dual channel device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and $\overline{\text{FR\_B\_TX\_EN}}$ is connected to the physical bus channel B. The dual channel system is shown in the following figure.

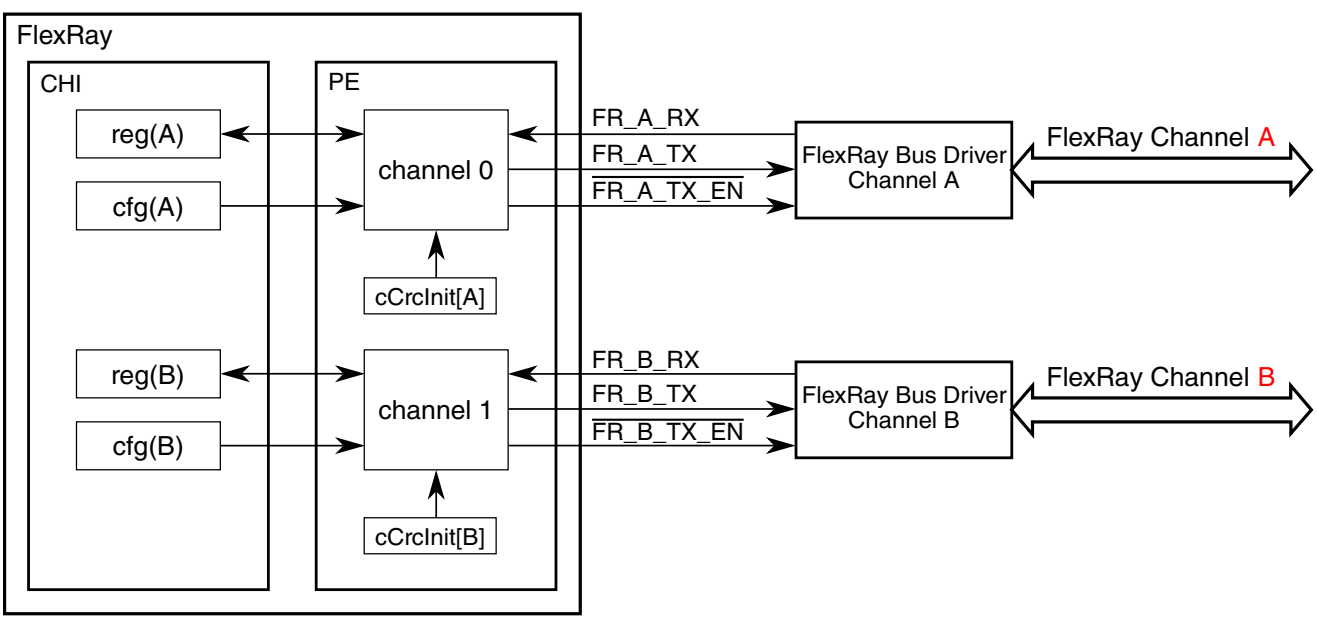


Figure 60-22. Dual Channel Device Mode

60.8.10.2 Single Channel Device Mode

The single channel device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ and can be connected to either the physical bus channel A (shown in the "Single Channel Device Mode (Channel A)" figure below) or the physical bus channel B (shown in the "Single Channel Device Mode (Channel B)" figure below).

If the device is configured as a single channel device by setting FR_MCR[SCM] to 1, only the internal channel A and the FlexRay Port A is used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay Channel A or FlexRay Channel B. The bit FR_MCR[CHA] must be set, if the FlexRay Port A is connected to a FlexRay Channel A. The bit FR_MCR[CHB] must be set if the FlexRay Port A is connected to a FlexRay Channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single channel device, the application can access and configure only the registers related to internal channel A.

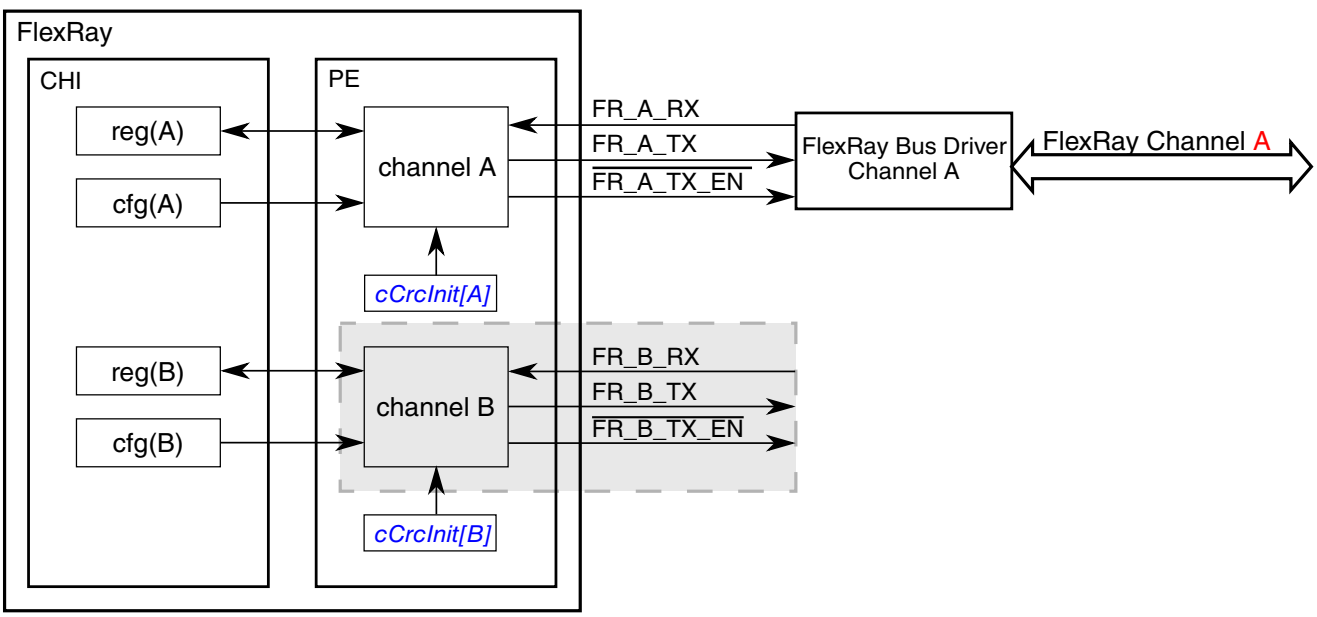


Figure 60-23. Single Channel Device Mode (Channel A)

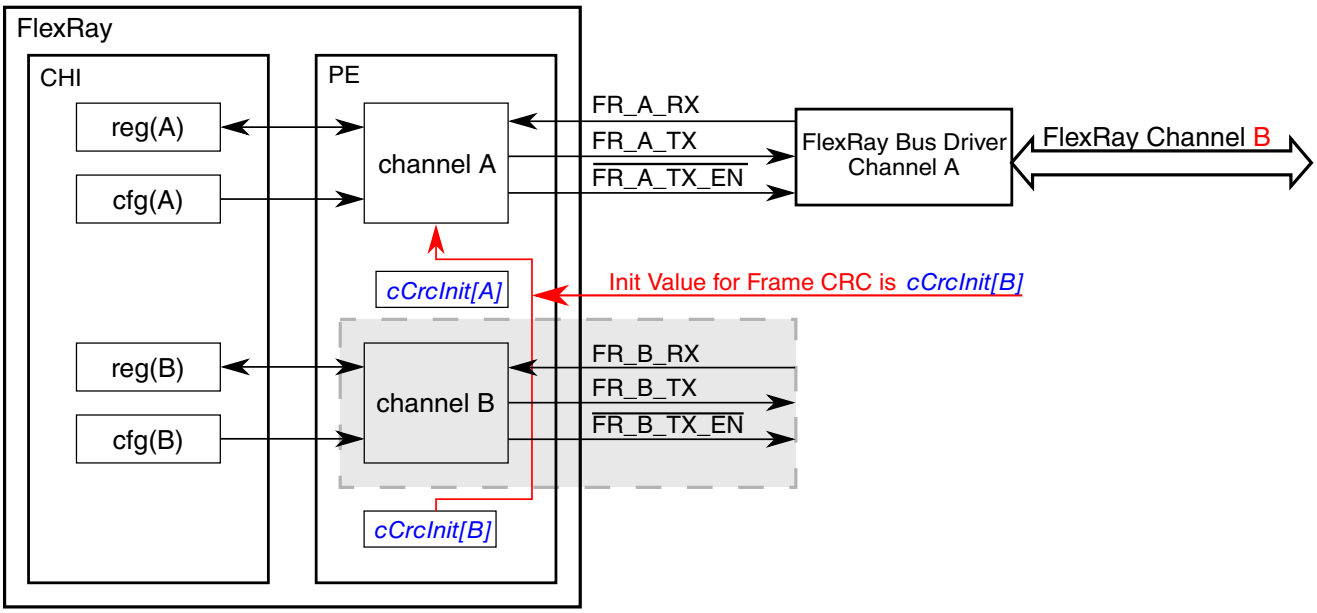


Figure 60-24. Single Channel Device Mode (Channel B)

60.8.11 External Clock Synchronization

The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in the "External Offset Correction Write and Application Timing" figure and the "External Rate Correction Write and Application Timing" figures below.

Note

The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n+1 shall be affect by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end of the static segment of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle 2n+1. If the value is not applied in cycle 2n+1, then the value will be applied in the cycle 2n+3. Refer to the following for timing details.

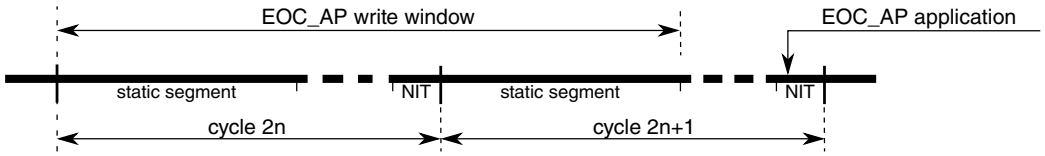


Figure 60-25. External Offset Correction Write and Application Timing

If the rate correction for the cycle pair [2n+2, 2n+3] shall be affect by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle pair [2n+2, 2n+3]. If the value is not applied for cycle pair [2n+2, 2n+3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to the following for details.

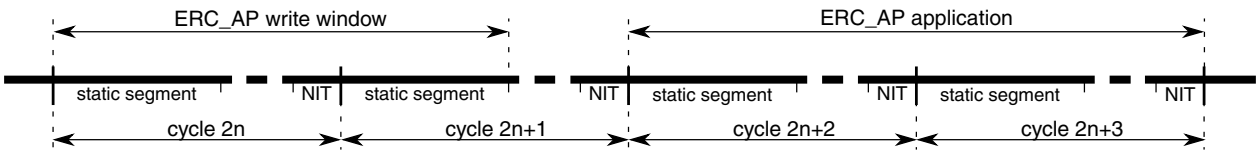


Figure 60-26. External Rate Correction Write and Application Timing

60.8.12 Sync Frame ID and Sync Frame Deviation Tables

The FlexRay protocol requires the provision of a snapshot of the Synchronization Frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note

Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

60.8.12.1 Sync Frame ID Table Content

The Sync Frame ID Table is a snapshot of the protocol related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

60.8.12.2 Sync Frame Deviation Table Content

The Sync Frame Deviation Table is a snapshot of the protocol related variable zsDev(id)(oe)(ch)!Value. Each Sync Frame Deviation Table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the Sync Frame ID Table.

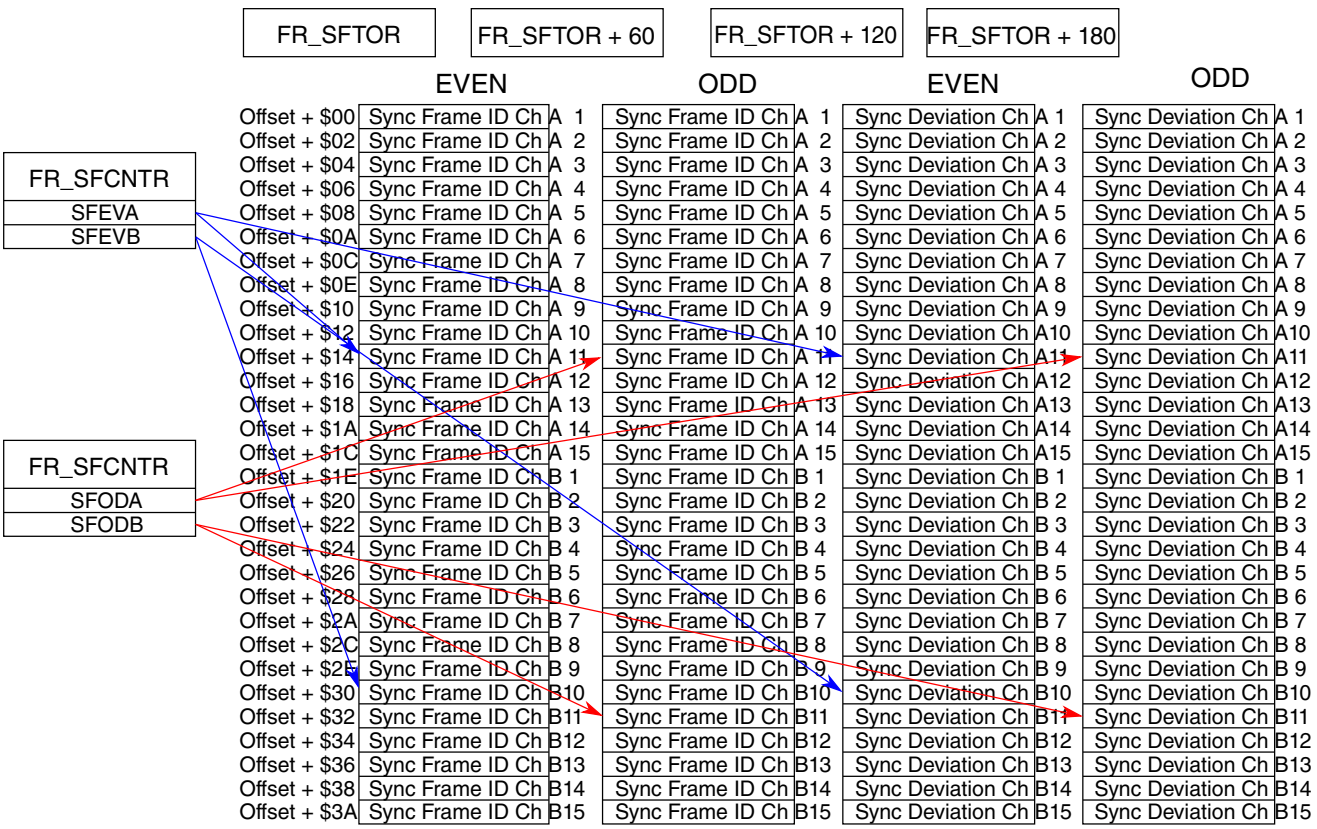


Figure 60-27. Sync Table Memory Layout

60.8.12.3 Sync Frame ID and Sync Frame Deviation Table Setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in Figure 60-27. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

60.8.12.4 Sync Frame ID and Sync Frame Deviation Table Generation

The application controls the generation process of the Sync Frame ID and Sync Frame Deviation Tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in the table below.

Table 60-53. Sync Frame Table Generation Modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No Sync Frame Table copy
0	0	1	Sync Frame ID Tables will be copied continuously
0	1	0	Reserved
0	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables will be copied continuously
1	0	0	No Sync Frame Table copy
1	0	1	Sync Frame ID Tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables for next even-odd-cycle pair will be copied

The Sync Frame Table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even cycle related tables after the start of the NIT of the next even cycle. The CC checks if the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. Once all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

To read the generated tables, the application must lock the tables to prevent the CC from updating these tables. The locking is initiated by writing a 1 to the even table lock trigger FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding

status information fields SFRA, SFRB in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in Figure 60-27.

After reading all the data from the locked tables, the application must unlock the table by writing to the even table lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

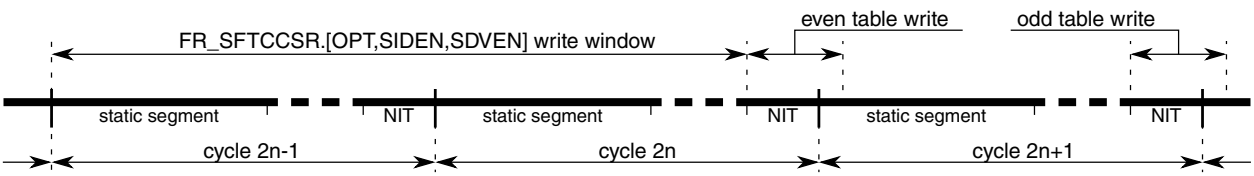


Figure 60-28. Sync Frame Table Trigger and Generation Timing

60.8.12.5 Sync Frame Table Access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in Figure 60-27. During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

60.8.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

60.8.13 MTS Generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol MTS in the symbol window on channel A or channel B.

The application can configure the set of communication cycles in which the MTS will be transmitted over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR\_PSR0[PROTSTATE] = POC:\ normalactive$$

Equation 34

$$FR\_MTSACRF[MTE] = 1$$

Equation 35

$$CYCCNT \,\&\, FR\_MTSACFR[CYCCNTMSK] = FR\_MTSACFR[CYCCNTVAL] \,\&\, FR\_MTSACFR[CYCCNTMSK]$$

Equation 36

The MTS is transmitted over channel B in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR\_MTSBCRF[MTE] = 1$$

Equation 37

$$CYCCNT \,\&\, FR\_MTSBCFR[CYCCNTMSK] = FR\_MTSBCFR[CYCCNTVAL] \,\&\, FR\_MTSBCFR[CYCCNTMSK]$$

Equation 38

60.8.14 Key Slot Transmission

Key slot assignment, transmission in POC:startup state, and transmission in POC:normal active state are discussed in this section.

60.8.14.1 Key Slot Assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2), otherwise no key slot is assigned.

60.8.14.2 Key Slot Transmission in POC:startup

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version *2.1 Rev A* .

60.8.14.3 Key Slot Transmission in POC:normal active

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in the following table is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see Message Transmission). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see Null Frame Transmission).

Table 60-54. Key Slot Frame Type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	normal frame
0	1	normal frame1[1]
1	0	sync frame
1	1	startup frame

1. The frame transmitted has an semantically incorrect header and will be detected as an invalid frame at the receiver.

60.8.15 Sync Frame Filtering

Each received synchronization frame must pass the Sync Frame Acceptance Filter and the Sync Frame Rejection Filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled, i.e. the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared, all received synchronization

frames are considered for clock synchronization. If a received synchronization frame did not pass at least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

60.8.15.1 Sync Frame Acceptance Filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if the following equations evaluates to true.

$$FR\_MCR[SFFE] = 0$$

Equation 39

$$FID \,\&\, FR\_SFIDAFMR[FMSK] = FR\_SFIDAFVR[FVAL] \,\&\, FR\_SFIDAFMR[FMSK]$$

Equation 40

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

60.8.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if the following equations evaluates to true.

$$FR\_MCR[SFFE] = 0$$

Equation 41

$$FID \neq FR\_SFIDRFR[SYNFRID]$$

Equation 42

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

60.8.16 Strobe Signal Support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in the Strobe Signal Mapping table, located in the Register Descriptions section.

60.8.16.1 Strobe Signal Assignment

Each of the strobe signals listed in the Strobe Signal Mapping table, located in the Register Descriptions section, can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number N, the application must execute the following sequence.

1. Write to FR_STBSCR with WMD = 1 and SEL = N. (updates SEL field only)
2. Read STBCSR.

   The SEL field provides N and the ENB and STBPSEL fields provides the settings for signal N.

60.8.16.2 Strobe Signal Timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay Bus. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a negative clock offset. An example waveform is given in the following figure.

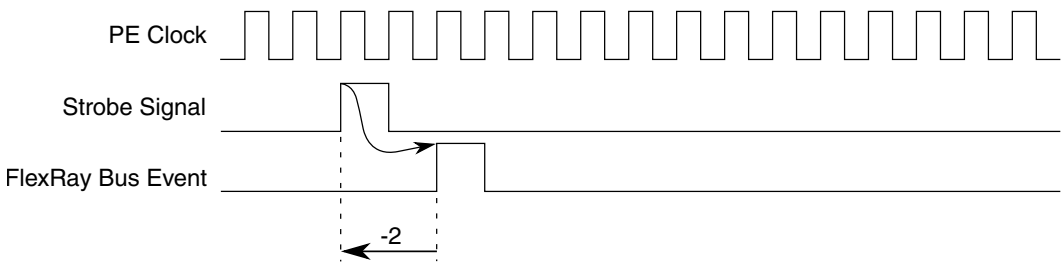


Figure 60-29. Strobe Signal Timing (type = pulse, clk_offset = -2)

Other signals refer to events that occurred on the FlexRay Bus some cycles before the strobe signal is changed. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a positive clock offset. An example waveform is given in the figure below.

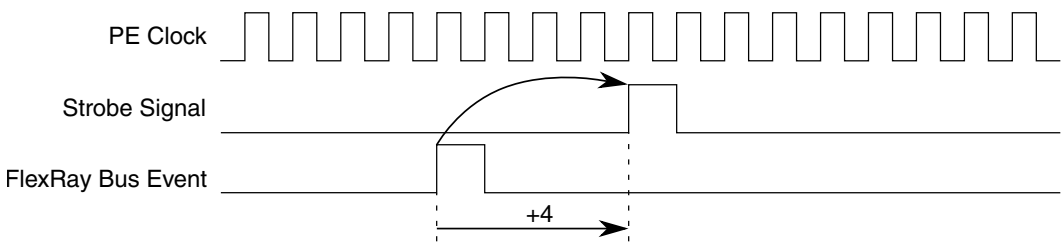


Figure 60-30. Strobe Signal Timing (type = pulse, clk_offset = +4)

60.8.17 Timer Support

The CC provides two timers, which run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In the non-repetitive mode, timer stops if it expires. In repetitive mode, timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

60.8.17.1 Absolute Timer T1

The absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if the following equations are fulfilled.

$$CYCTR[CTCCNT] \& FR\_TI1CYSR[T1\_CYC\_MSK] = FR\_TI1CYSR[T1\_CYC\_VAL] \& FR\_TI1CYSR[T1\_CYC\_MSK]$$

Equation 43

$$FR\_MTCTR[MTCT] = FR\_TI1MTOR[T1\_MTOFFSET]$$

Equation 44

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

60.8.17.2 Absolute / Relative Timer T2

The timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

60.8.17.2.1 Absolute Timer T2

If timer T2 is configured as an absolute timer, it has the same functionality timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. On expiration of timer T2, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

60.8.17.2.2 Relative Timer T2

If the timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set, when the programmed amount of macroticks MT[31:0], defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1), has expired since the trigger or restart of timer 2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.

60.8.18 Slot Status Monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per channel base. The content of the slot status vector is described in the "Slot Status Content" table below. The PE provides the slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in the below figure.

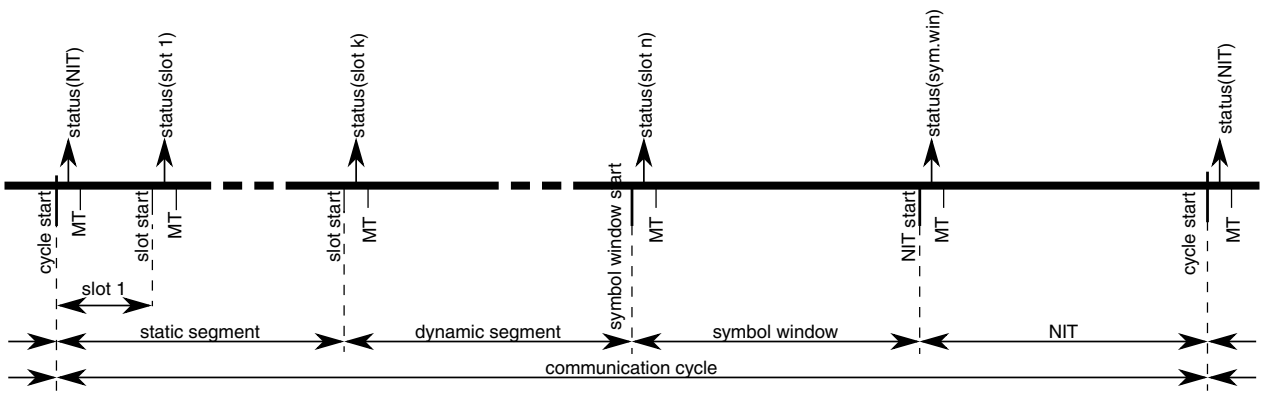


Figure 60-31. Slot Status Vector Update

Note

The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 60-55. Slot Status Content

	Status Content
static /dynamic Slot	**slot related status**
	vSS!ValidFrame - valid frame received
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!ContentError - content error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	for slots in which the module transmits:
	vSS!TxConflict - reception ongoing while transmission starts
	for slots in which the module does not transmit:
	vSS!TxConflict - reception ongoing while transmission starts
	first valid - channel that has received the first valid frame
	received frame related status
	extracted from
	a)header of valid frame, if vSS!ValidFrame = 1

Table continues on the next page...

Table 60-55. Slot Status Content (continued)

	Status Content
	b) last received header, if vSS!ValidFrame = 0
	c) set to 0, if nothing was received
	vRF!Header!NFIndicator - Null Frame Indicator (0 for null frame)
	vRF!Header!SuFIndicator - Startup Frame Indicator
	vRF!Header!SyFIndicator - Sync Frame Indicator
Symbol Window	**window related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - reception ongoing while transmission starts
	received symbol related status
	vSS!ValidMTS - valid Media Test Access Symbol received
	received frame related status
	see static/dynamic slot
NIT	**NIT related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - always 0
	received frame related status
	see static/dynamic slot

60.8.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR), incremented by one, if at least one of four slot status error bits, *vSS!SyntaxError, vSS! ContentError, vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap round after they have reached the maximum value.

60.8.18.2 Protocol Status Registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the Network Idle Time NIT and the Symbol Window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

60.8.18.3 Slot Status Registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status related and received frame / symbol related status information, as given in Table 60-55, except of the *first valid* indicator for non-transmission slots.

60.8.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in the figure below.

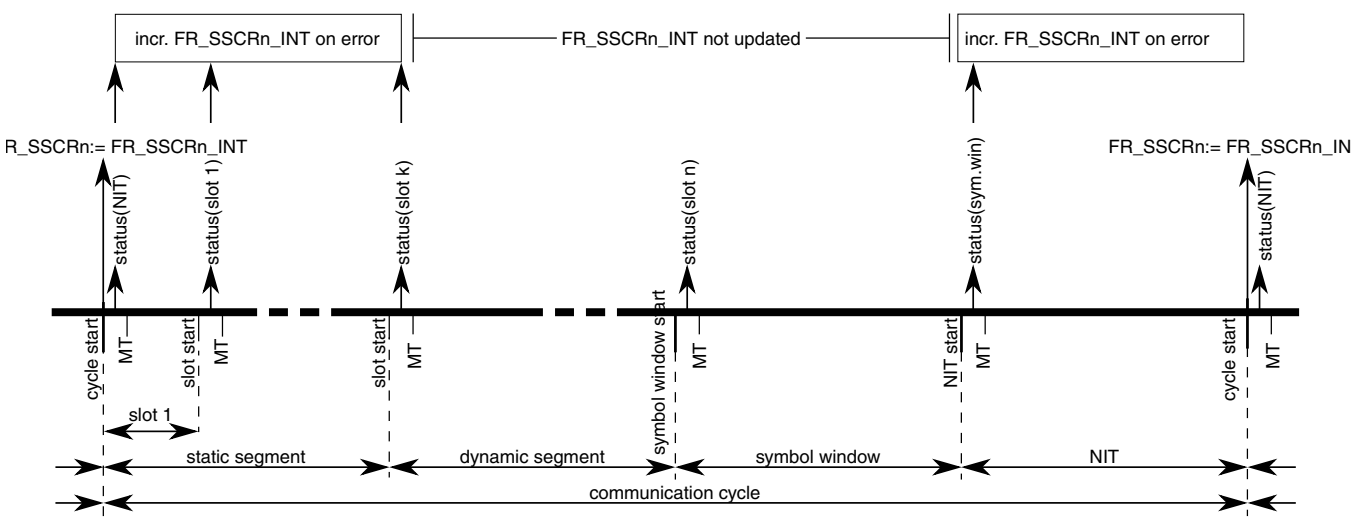


Figure 60-32. Slot Status Counting and FR_SSCRn Update

The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCRn register reflects, in cycle n, the status of the NIT of cycle n-2, and the status of all static slots and the symbol window of cycle n-1.

The increment condition for each slot status counter consists of two parts, the frame related condition part and the slot related condition part. The internal slot status counter FR_SSCRn_INT is incremented if at least one of the conditions is fulfilled:

1. frame related condition:

   (FR_SSCCRn[VFR] | FR_SSCCRn[SYF] | FR_SSCCRn[NUF] | FR_SSCCRn[SUF]) // count on frame condition

   = 1;

   and

   ((~FR_SSCCRn[VFR] | vSS!ValidFrame) & // valid frame restriction

   (~FR_SSCCRn[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction

   (~FR_SSCCRn[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction

   (~FR_SSCCRn[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction

   = 1;

   Note

   The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus it is required to set the VFR always, whenever count on frame condition is used.

   slot related condition:

   ((FR_SSCCRn[STATUSMASK[3]] & vSS!SyntaxError) | // increment on syntax error

   (FR_SSCCRn[STATUSMASK[2]] & vSS!ContentError) | // increment on content error

   (FR_SSCCRn[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation

   (FR_SSCCRn[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict

= 1;

If the slot status counter is in single cycle mode, i.e. FR_SSCCRn[MCY] = 0, the internal slot status counter FR_SSCRn_INT is reset at each cycle start. If the slot status counter is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset and incremented, until the maximum value is reached.

60.8.18.5 Message Buffer Slot Status Field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in Table 60-55 for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update Sections in Individual Message Buffer Configuration Data.

60.8.19 System Bus Access

This section provides a description of the system bus accesses failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in System Bus Access Failure Types.

The behavior of the CC after the occurrence of a system bus access failure is described in System Bus Access Failure Response.

60.8.19.1 System Bus Access Failure Types

This section describes the two types of system bus access failures.

60.8.19.1.1 System Bus Illegal Address Access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions which are treated as a system bus illegal address access:

- The system bus subsystem detects an CC access to an illegal system memory address.

- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see CHI LRAM Error Response after CC Read).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

60.8.19.1.2 System Bus Access Timeout

A system bus access timout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see Configure System Memory Access Time-Out Register (FR_SYMATOR)).

If a system bus access timout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

60.8.19.2 System Bus Access Failure Response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

60.8.19.2.1 Continue after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is 0, the CC will continue its operation after the occurrence of the system bus access failure, but will not generate any system bus accesses until the start of the next communication cycle.Since no data are read from or written to the FlexRay memory area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated with the remaining header and frame data are replaced by all zeros. Depending on the point in time this can affect the PPI bit, the Header CRC, the Payload Length in case of an dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted and received, except for the key slot, where a sync or startup null-frame is transmitted, if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

60.8.19.2.2 Freeze after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is set to 1, the CC will go into the freeze mode immediately after the occurrence of one of the system bus access failures.

60.8.20 Interrupt Support

The CC provides 172 individual interrupt sources and five combined interrupt sources.

60.8.20.1 Individual Interrupt Sources

Message buffer interrupts are discussed in this section.

60.8.20.1.1 Message Buffer Interrupts

The CC provides 128 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSRn[MBIF] and an interrupt enable bit FR_MBCCSRn[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

60.8.20.1.2 FIFO Interrupts

The CC provides 2 FIFO interrupt sources.

Each of the 2 FIFO provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding Receive FIFO fill level exceeds the defined watermark.

60.8.20.1.3 Wakeup Interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

60.8.20.1.4 Protocol Interrupts

The CC provides 25 interrupt sources for protocol related events. For details, see Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

60.8.20.1.5 CHI Interrupts

The CC provides 16 interrupt sources for CHI related error events. For details, see CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

60.8.20.2 Combined Interrupt Sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

60.8.20.2.1 Receive Message Buffer Interrupt

The Receive Message Buffer Interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

60.8.20.2.2 Transmit Message Buffer Interrupt

The Transmit Message Buffer Interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

60.8.20.2.3 Protocol Interrupt

The Protocol Interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

60.8.20.2.4 CHI Interrupt

The CHI Interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

60.8.20.2.5 Module Interrupt

The Module Interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

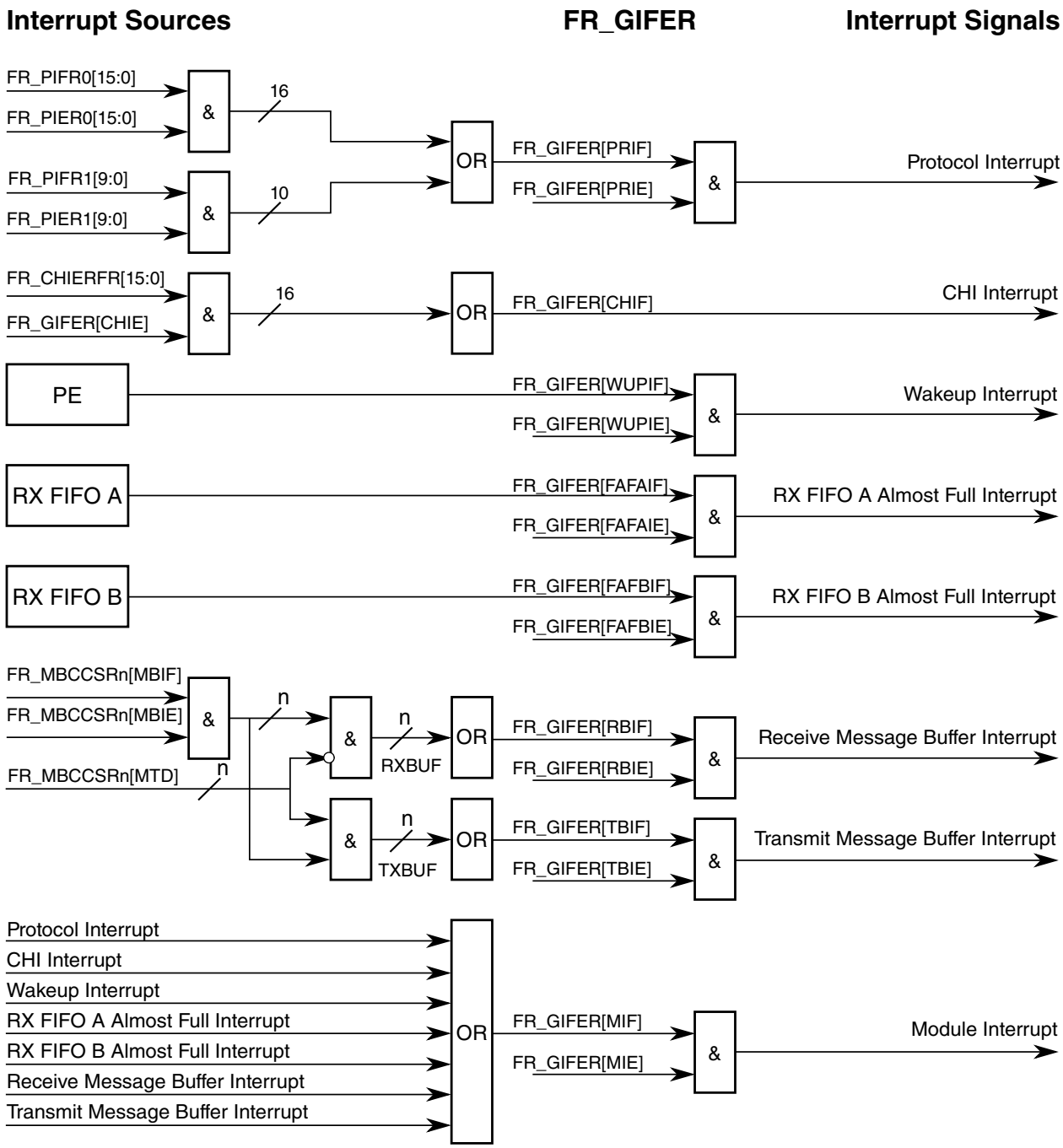


Figure 60-33. Scheme of FR_GIFER interrupt signal generation

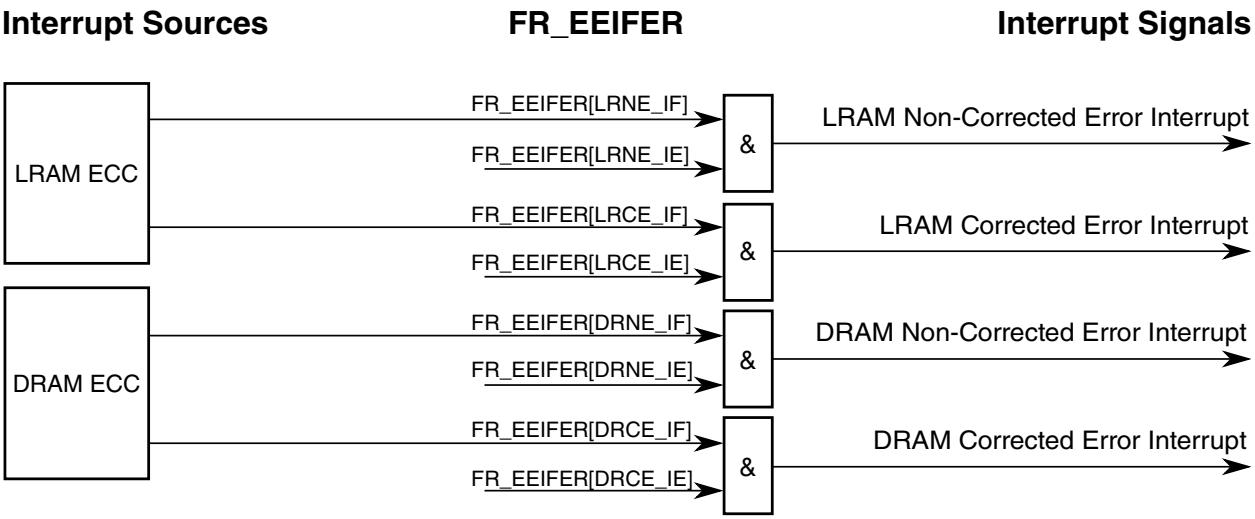


Figure 60-34. Scheme of FR_EEIFER interrupt signal generation

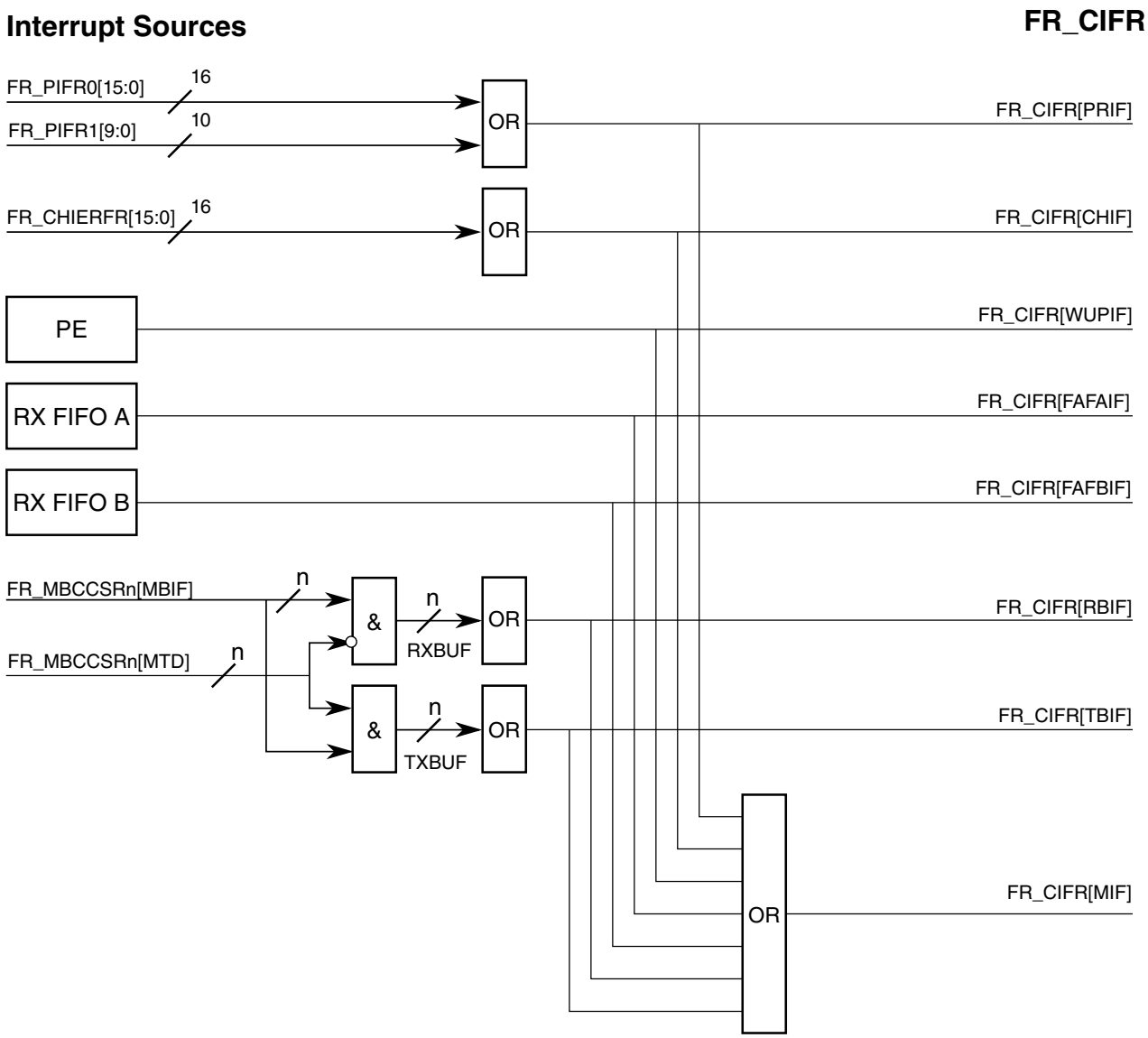


Figure 60-35. Scheme of FR_CIFR flags generation

60.8.21 Lower Bit Rate Support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset*. The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings and related protocol parameter values are shown in the "FlexRay Channel Bit Rate Control" table below.

Table 60-56. FlexRay Channel Bit Rate Control

FlexRay Channel Bit Rate [Mbit/s]	FR_MCR[BITRATE]	*pdMicrotick [ns]*	*gdSampleClockPeriod [ns]*	*pSamplesPerMicrotick*	*cSamplesPerBit*	*cStrobeOffset*
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note

The bit rate of 8 Mbit/s is not defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

60.8.22 PE Data Memory (PE DRAM)

The PE Data Memory (PE DRAM) is 128 word, 16-bit wide memory with byte access, which contains the program data of the PE internal CPU. The PE DRAM is divided into two banks, 8-bit each. The memory data [7:0] are assigned to BANK0, the memory data [15:8] are assigned to BANK1.

Table 60-57. PE DRAM Layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via PE DRAM Access Register (FR_PEDRAR) and PE DRAM Access Register (FR_PEDRAR). This functionality is used to check the memory error detection.

60.8.22.1 PE DRAM Read Access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;

   // INST=0x5; ADDR=070

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA];

   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25us).

60.8.22.2 PE DRAM Write Access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;

   // write value to be written into data register

2. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

3. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

4. val = FR_PEDRDR[DATA];

   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may causes an response delay with a maximum of 1000 PE clock cycle (25us).

If the conditions given in PE DRAM Write Access Limitations are fulfilled, the data provided in PE DRAM Access Regsiter (FR_PEDRAR) are written into the PE DRAM, read back in the next clock cycle and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data are not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

60.8.22.3 PE DRAM Write Access Limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not *in POC:default config* state, only PE DRAM address 0x70 is writable.

60.8.23 CHI Lookup-Table Memory (CHI LRAM)

The CHI Lookup-Table Memory (CHI LRAM) is an CHI internal memory which contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or 6 data field offsets are contained in one memory row. The CHI LRAM is divided into 6 memory BANKs.

Table 60-58. CHI LRAM Layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
…						
0x3F	FR_MBIDXR127	FR_MBFIDR127	FR_MBCCFR127	FR_MBIDXR126	FR_MBFIDR126	FR_MBCCFR126
0x40	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
…						
0x55	FR_MBDOR131	FR_MBDOR130	FR_MBDOR129	FR_MBDOR128	FR_MBDOR127	FR_MBDOR126
0x56	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

60.8.23.1 CHI LRAM Read and Write Access

The CHI LRAM is accessed by the application via regular register read and write accesses.

60.8.24 Memory Content Fault Detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses an standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED) . The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

60.8.24.1 Memory Error Types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as an special case. The FlexRay module detects and indicates the memory errors as shown in the "Detected Memory Error Types" table below. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row, and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error free bank has been read.

Table 60-59. Detected Memory Error Types

Memory	Priority	Memory Data	Indication
CHI LRAM	0 (highest)	All Zero's	Non-Corrected Error
PE DRAM			Non-Corrected Error
CHI LRAM		All One's	Non-Corrected Error
PE DRAM			
CHI LRAM	1 (lowest)	One Bit Flipped	Non-Corrected Error
PE DRAM			Corrected Error

Table continues on the next page...

Table 60-59. Detected Memory Error Types (continued)

Memory	Priority	Memory Data	Indication
CHI LRAM		Two Bits Flipped	Non-Corrected Error
PE DRAM			
CHI LRAM		Three or more Bits Flipped	one out of {No error, Non-Corrected Error}, defined by coding given in CHI LRAM Checkbits and CHI LRAM Syndrome
PE DRAM			one out of {No error, Corrected Error, Non-Corrected Error}, defined by coding given in PE DRAM Checkbits and PE DRAM Syndrome

60.8.24.2 Memory Error Reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories exists two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- *If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.*
- *If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.*

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to above the Error indications (corrected / non corrected) along with the failing address is sent out from the module for external error logging and corrective actions by the Software.

60.8.24.2.1 PE DRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in Table 60-59. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 60-60. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4[1]	X	X	X	X	X	X	X	X	X	X	X	X
3[2]	-	-	-	-	X	X	X	X	-	-	-	-
2	-	-	-	-	X	-	-	-	X	X	X	-
1	-	-	-	-	-	X	X	-	X	X	-	X
0	-	-	-	-	-	X	-	X	X	-	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is a even number of 1's in columns with X.
2. The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is a odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words and written into the memory.

60.8.24.2.2 PE DRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in the following table.

Table 60-61. FR_EERCR[CODE] PE DRAM Syndrome Coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (Never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected Error (Parity Bit 0)
0x0	0x2	Corrected Error (Parity Bit 1)
0x0	0x3	Corrected Error (Data Bit 0)
0x0	0x4	Corrected Error (Parity Bit 2)
0x0	0x5	Corrected Error (Data Bit 1)
0x0	0x6	Corrected Error (Data Bit 2)
0x0	0x7	Corrected Error (Data Bit 3)
0x0	0x8	Corrected Error (Parity Bit 3)
0x0	0x9	Corrected Error (Data Bit 4)

Table continues on the next page...

Table 60-61. FR_EERCR[CODE] PE DRAM Syndrome Coding (continued)

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x0	0xA	Corrected Error (Data Bit 5)
0x0	0xB	Corrected Error (Data Bit 6)
0x0	0xC	Corrected Error (Data Bit 7)
0x0	0xD-0xF	Non-Corrected Error
0x1	0x1-0xF	Non-Corrected Error

60.8.24.2.3 CHI LRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the table below. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 60-62. CHI LRAM checkbits coding

CODE[1]	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	X	X	X	X	X	X	X	-	-	-	-
2	X	X	-	-	-	X	X	X	X	-	-	-	X	X	X	-
1	-	-	X	X	-	X	X	-	-	X	X	-	X	X	-	X
0	X	-	X	-	X	X	-	X	-	X	-	X	X	-	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is a odd number of 1's in all columns with X.

60.8.24.2.4 CHI LRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the next table.

Table 60-63. FR_EERCR[CODE] CHI LRAM Syndrome Coding

FR_EERCR[CODE]	Description
0x00	No Error (Never appears in error report registers)
0x01-0x1F	Non Corrected Error

60.8.24.3 Memory Error Response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case of the detection of a *corrected* memory error, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

60.8.24.3.1 CHI LRAM Error Response after CC Read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access the error response of the module depends from LRAM location where the error occurred.

- *If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.*

  If the affected message buffer is a tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is a rx message buffer, no frame will be received to this message buffer in the next slot.

- *If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs w.r.t. payload access is performed for the complete frame.*

60.8.24.3.2 CHI LRAM Error Response after Application Read

The application can read the content of the CHI LRAM via reading the FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn, FR_MBDORn, and FR_LEETRn registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read and continues its normal operation.

60.8.24.3.3 PE DRAM Error Response after CC Read

If the CC detects an non-corrected memory error during internal read of program data which is contained in PE DRAM, this is considered as an fatal protocol error and the module enters the protocol freeze state immediately.

60.8.24.3.4 PE DRAM Error Response after Application Read in *POC:default config* state

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered as an fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

60.8.24.3.5 PE DRAM Error Response after Application Read out of *POC:default config*

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address, and the protocol is not in the *POC:default config* state, this error is not considered as an fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

60.8.25 Memory Fault Injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

The error injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).

When the error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

60.8.25.1 CHI LRAM Error Injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in Table 60-58.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see Memory Fault Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality

2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode

3. FR_EEIAR[MID]:= 1;

   // select CHI LRAM for error injection

4. FR_EEIAR[BANK]:= I_BANK;

   // select bank for error injection; I_BANK = {0,1,2,3,4,5}

5. FR_EEIAR[ADDR]:= I_ADDR;

   // select address for error injection; I_ADDR <= 0x 56

6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern

7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern

8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;

If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;

If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;

If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;

If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;

If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;

// write DATA to the defined injection bank and injection address (see Table 60-58).

60.8.25.2 PE DRAM Error Injection

The following sequence describes an memory error injection sequence for the PE DRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see PE DRAM Error Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality

2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode

3. FR_EEIAR[MID]:= 0;

   // select PE DRAM for error injection

4. FR_EEIAR[BANK]:= I_BANK;

   // define bank for error injection; I_BANK = {0,1}

5. FR_EEIAR[ADDR]:= I_ADDR;

   // define address for error injection; I_ADDR <= 0x7F

6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern

7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern

8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access (e.g. I_ADDR=0x70):

1. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA]; |

   // get read back PE DRAM data

Note

The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

60.9 Application Information

Module operation is discussed in this section.

60.9.1 Module Configuration

This section describes essential parts of the module configuration.

60.9.1.1 Configure System Memory Access Time-Out Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill this equation[1] :

$$0 \le \mathrm{SYMATOR}[\mathrm{TIMEOUT}] \le \lfloor 0.45 \cdot f_{\mathrm{CHI}} - 8 \rfloor$$

Equation 45

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in the table below.

Table 60-64. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
>= 18 MHz	0	>= 100 MHz	<= 37
>= 23 MHz	<= 2	>= 120 MHz	<= 46
>= 27 MHz	<= 4	>= 140 MHz	<= 55
>= 32 MHz	<= 6	>= 160 MHz	<= 64
>= 60 MHz	<= 19	>= 180 MHz	<= 73
>= 80 MHz	<= 28	>= 200 MHz	<= 82

60.9.1.1.1 System Bus Wait State Constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single channel configurations and if the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0) no timeout will be detected if less than 2 × SYMATOR[TIMEOUT]+1 wait states will be seen on the system bus for each system bus access.

For dual channel configurations, or if the sync frame table generation functionality is used, no timeout will be detected if less than SYMATOR[TIMEOUT]-1 wait states will be seen on the system bus for each system bus access.

60.9.1.2 Configure Data Field Offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDORn) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers which are used.

When the module is enabled, the initilization value of the FR_MBDORn[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see System Bus Illegal Address Access).

60.9.2 Initialization Sequence

This section describes the required steps to initialize the CC. The first subsection describes the steps required after a module reset, the second section describes the steps required after preceding shutdown of the CC.

60.9.2.1 Module Initialization

This section describes the module related initialization steps after a system reset.

1. Configure CC.
    a. configure the control bits in the Module Configuration Register (FR_MCR)
    b. configure the system memory base address in System Memory Base Address Register (FR_SYMBADR)
2. Enable the CC.
    a. write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

The CC now enters the Normal Mode. The application can commence with the protocol initialization described in Protocol Initialization.

60.9.2.2 Protocol Initialization

This section describes the protocol related initialization steps.

1. Configure the Protocol Engine.
    a. issue CONFIG command via Protocol Operation Control Register (FR_POCR)
    b. wait for *POC:config* in Protocol Status Register 0 (FR_PSR0)
    c. configure the FR_PCR0,…, FR_PCR30 registers to set all protocol parameters
2. Configure the Message Buffers and FIFOs.
    a. set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

b. define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR)

c. configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn), Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), Message Buffer Index Registers (FR_MBIDXRn)

d. configure the FIFOs

e. issue CONFIG_COMPLETE command via Protocol Operation Control Register (FR_POCR)

f. wait for *POC:ready* in Protocol Status Register 0 (FR_PSR0)

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

60.9.2.3 CHI LRAM Initialization

The initialization of the CHI LRAM is performed by the CC when it leaves the Disabled Mode. The unitization runs for 87 CHI clock cycles. All fields in the FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBDORn, and LEETRn registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

60.9.2.4 PE DRAM Initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs, and will delay the state transition from *POC:default config* into *POC:config*.

60.9.3 Memory Fault Injection out of POC:default config

This section provides information for application driven memory fault injection out if *POC:default config*. The CC provides means to inject memory faults from the application without any impacts to the internal protocol operation of the CC.

60.9.3.1 CHI LRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETRn). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in CHI LRAM Error Injection.

60.9.3.2 PE DRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only one PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM location to inject memory errors into the PE DRAM. The injection sequence is described in PE DRAM Error Injection.

60.9.4 Shut Down Sequence

This section describes a secure shut down sequence to stop the CC gracefully. The main targets of this sequence are

- *finish all ongoing reception and transmission*
- *do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication*

For a graceful shutdown the application shall perform the following tasks:

1. Disable all enabled message buffers.
   a. repeatedly write 1 to FR_MBCCSRn[EDT] until FR_MBCCSRn[EDS] == 0.
2. Stop Protocol Engine.
   a. issue HALT command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0)

60.9.5 Number of Usable Message Buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[1].

Additional constraints for the minimum CHI clock frequency are given in Controller Host Interface Clocking.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of slot and must be finished before the start of the next slot.

The shortest FlexRay slot is a corrected empty dynamic slot. An corrected empty dynamic slot is a mini-slot and consists of gdMinislot corrected macroticks with a duration of *gdMacrotick*. The minimum duration of an corrected macrotick is $gdMacrotick_{min}$ = 39 µT. This results in a minimum length of an correct slot

$$\Delta_{slotmin} = 39 \bullet pdMicrotick \bullet gdMinislot$$

Equation 46

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

$$\Delta_{search} = \frac{1}{f_{chi}} \bullet (FR\_MBSSUTR[LAST\_MB\_UTIL] + 27)$$

Equation 47

The message buffer search must be finished within one slot which requires fulfillment of this equation::

$$\Delta_{search} \leq \Delta_{slotmin}$$

Equation 48

This results in the formula given below which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

$$f_{chi} \geq \frac{(FR\_MBSSUTR[LAST\_MB\_UTIL]+27)}{39 \cdot pdMicrotick \cdot gdMinislot}$$

Equation 49

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 127 is given in the following table.

Table 60-65. Minimum fchi [MHz] examples (128 message buffers used)

pdMicrotick [ns]	*gdMinislot*					
	2	**3**	**4**	**5**	**6**	**7**
25.0	79.5	53	39.8	31.8	26.5	22.8
50.0	39.8	26.5	19.9	15.9	13.3	11.4

Note

If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to the CHI Error Flag Register (FR_CHIERFR) for details.

60.9.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of the FR_POCR Field Descriptions table, located in the Register Descriptions section, by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in the next table. If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by the table below. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 60-66. Protocol Control Command Priorities

Protocol Command	Priority	Interrupted By	Cleared and Terminated By
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLET, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE,fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE,fatal protocol error

60.9.7 Message Buffer Search On Simple Message Buffer Configuration

This sections describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

60.9.7.1 Simple Message Buffer Configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *S*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has following configuration

Table 60-67. Transmit Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	transmit buffer

Table continues on the next page...

Table 60-67. Transmit Buffer Configuration (continued)

Register	Field	Value	Description
FR_MBCCFR*t*	MTM	0	event transition mode
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000011	cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	*S*	assigned to slot S

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number r and has following configuration

Table 60-68. Receive Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*r*	MTD	0	receive buffer
FR_MBCCFR*r*	MTM	-	n/a
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000001	cycle set = {2n} = {0,2,4,6,…}
	CCFVAL	000000	
FR_MBFIDR*r*	FID	*S*	subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSRt[EDS] = 1 and FR_MBCCSRr[EDS] = 1)

Note

The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

60.9.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *S* in the *static* segment.

The configuration of a transmit buffer for a static slot *S* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *S*. In any case, the result of the message buffer search will be the transmit message buffer *t*. The receive message buffer *r* will not be found, no reception is possible.

60.9.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *S* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

60.9.7.3.1 Transmit Data Not Available

If transmit data are *not available*, i.e. the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 and/or locked FR_MBCCSR*t*[LCKS]=1,

1. for the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in the following figure.

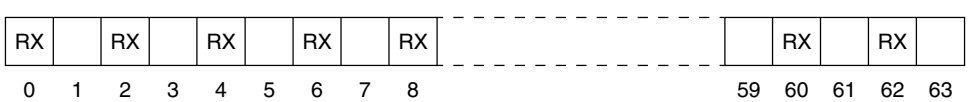


Figure 60-36. Transmit Data Not Available

60.9.7.3.2 Transmit Data Available

If transmit data are *available*, i.e. the transmit buffer is committed FR_MBCCSR*t*[CMT]=1 and not locked FR_MBCCSR*t*[LCKS]=0,

1. for the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in the following figure.

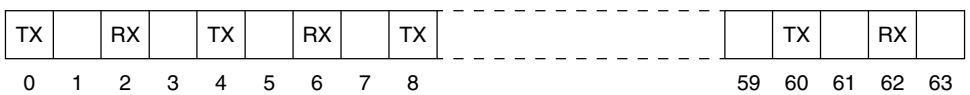


Figure 60-37. Transmit Data Available

Chapter 61
Ethernet MAC (ENET)

61.1 Chip-specific ENET information

61.1.1 ENET Register accesses

Always configure ENET registers before configuring RGMII pads. The ENET registers cannot be accessed in user mode until at least one supervisor-mode access is made to the ENET module and then the register protection user access allowed bit (REG_PROT_GCR[UAA]) is programmed. See the Register Protection chapter for more details on user access enabling of protected registers.

61.1.2 IEEE1588 (Ethernet PTP) to eTimer synchronization

The following tables lists IEEE1588 Ethernet timer input events and output events.

NOTE

For details on the PADS listed in the following tables, refer the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document.

Table 61-1. IEEE 1588 Ethernet timer input connectivity

Input module	Input signal	Driving module	Driver signal
ENET	TIMER0	eTIMER1	TIMER7
ENET	TIMER1	From PAD	-
ENET	TIMER2	CTE	RCS
ENET	TIMER3	CTE	RFS

Table 61-2. IEEE 1588 Ethernet timer output connectivity

Output module	Output signal	Target module	Target signal
ENET	TIMER0	To PAD	-
ENET	TIMER1	To PAD	-
ENET	TIMER2	To PAD	-
ENET	TIMER3	eTIMER1, eTIMER2	AUX_TMR6

61.1.3 ENET Interrupt mapping

Table 61-3. ENET Interrupt mapping

Interrupt name	Description	Interrupt slot
GROUP1		
mac0_ts_timer	MAC 0 Periodic Timer Overflow	216
mac0_ts_avail	MAC 0 Time Stamp Available	
GROUP2		
mac0_tx_buffer	MAC 0 Transmit Buffer Done	217
mac0_tx_frame	MAC 0 Transmit Frame Done	
GROUP3		
mac0_rx_buffer	MAC 0 Receive Buffer Done	218
mac0_rx_frame	MAC 0 Receive Frame Done	
GROUP4		
mac0_wake_async	MAC 0 Wakeup Request - asynchronous	219
mac0_wake_sync	MAC 0 Wakeup Request (sync)	
mac0_rx_err	MAC 0 Payload Receive Error	
mac0_tx_under	MAC 0 Transmit FIFO Underrun	
mac0_coll_retry	MAC 0 Collision Retry Limit	
mac0_coll_late	MAC 0 Late Collision	
mac0_bus_err	MAC 0 Ethernet Bus Error	
mac0_xfr_done	MAC 0 MII Data Transfer Done	
mac0_gra_stop	MAC 0 Graceful Stop	
mac0_tx_babb	MAC 0 Babbling Transmit Error	
mac0_rx_babb	MAC 0 Babbling Receive Error	
GROUP5		
mac0_timer_async	MAC0 1588 Timer Interrupt - asynchronous	220
mac0_timer_sync	MAC0 1588 Timer Interrupt - synchronous	

61.1.4 Overview

This chapter will be made available in the next release of this reference manual.

61.2 Introduction

The MAC-NET core, in conjunction with a 10/100/1000[1]-Mbit/s MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP, and ICMP, providing wire speed services to client applications.

61.3 Overview

The core implements a triple-speed 10/100/1000[1]-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s and full-duplex gigabit[1] Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

1. Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.

61.3.1 Features

61.3.1.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s and gigabit[2] operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Supports gigabit[2] full-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
  - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
  - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
  - a 2-bit Reduced MII (RMII) operating at 50 MHz.
- Simple 64-Bit FIFO user-application interface
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
  - Implements automated pause frame (802.3 x31A) generation and termination, providing flow control without user application intervention
  - Pause quanta used to form pause frames — dynamically programmable
  - Pause frame generation additionally controllable by user application offering flexible traffic flow control
  - Optional forwarding of received pause frames to the user application
  - Implements standard flow-control mechanism
- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable promiscuous mode support to omit MAC destination address checking on receive

2. Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.

- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- MDIO master interface for PHY device configuration and management supports two programmable MDIO base addresses, and standard (IEEE 802.3 Clause 22) and extended (Clause 45) MDIO frame formats
- Supports legacy FEC buffer descriptors
- Interrupt coalescing reduces the number of interrupts generated by the MAC, reducing CPU loading

61.3.1.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols
- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis
- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers
- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked

- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

61.3.1.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

61.3.2 Block diagram

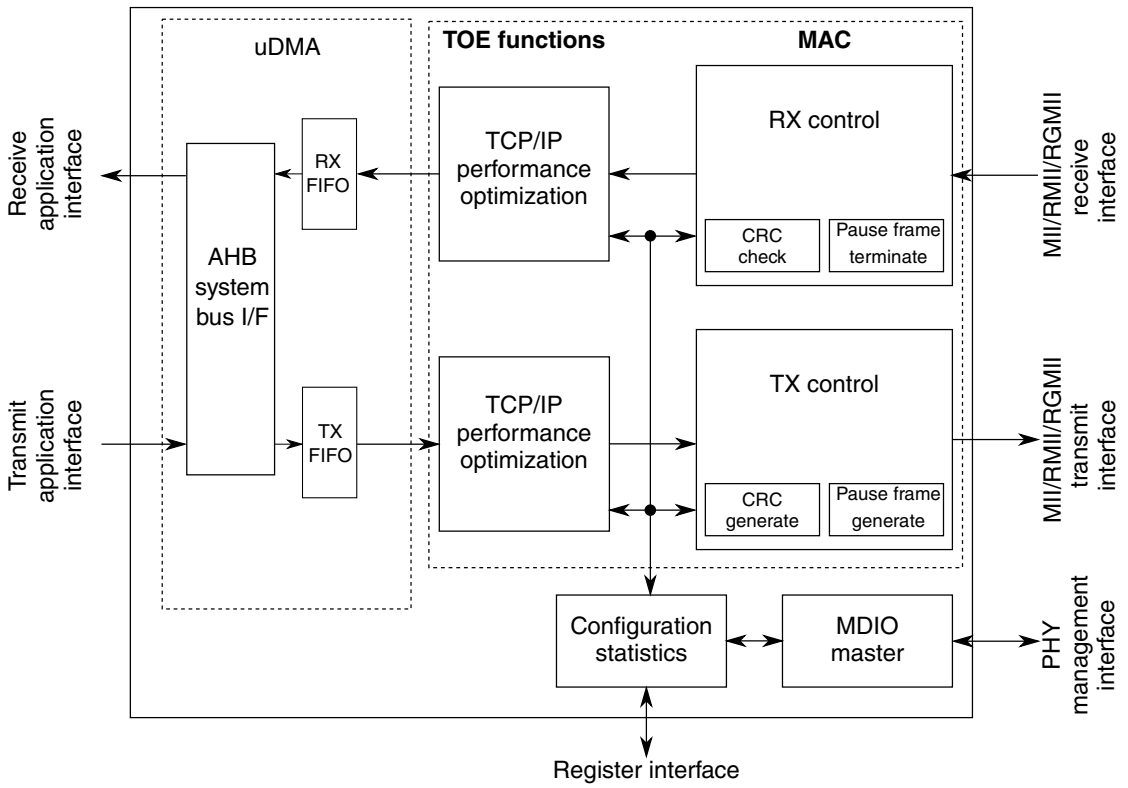


Figure 61-1. Ethernet MAC-NET core block diagram

61.4 External signal description

NOTE

The MII column pertains only to devices that support MII.

MII	RMII	RGMII	Description	I/O
MII_COL	—	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.	I
MII_CRS	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the RMII_CRS_DV pin.	I

Table continues on the next page...

MII	RMII	RGMII	Description	I/O
MII_MDC	RMII_MDC	RGMII_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	O
MII_MDIO	RMII_MDIO	RGMII_MDIO	Transfers control information between the external PHY and the media-access controller. Data is synchronous to MDC. This signal is an input after reset.	I/O
MII_RXCLK	—	RGMII_RXC	In MII mode, provides a timing reference for RXDV, RXD[3:0], and RXER. In RGMII mode, provides a timing reference for RGMII_RXD[3:0] and RGMII_RX_CTL.	I
MII_RXDV	RMII_CRS_DV	RGMII_RX_CTL	Asserting this input indicates the PHY has valid nibbles present on the MII. RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Asserting RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal. In RGMII mode, contains RXDV on the rising edge of RGMII_RXC, and a logical derivative of RXDV and RXERR (RXDV XOR RXERR) on the falling edge of RGMII_RXC.	I
MII_RXD[3:0]	RMII_RXD[1:0]	RGMII_RXD[3:0]	Contains the Ethernet input data transferred from the PHY to the media-access controller when RXDV is asserted.	I

Table continues on the next page...

MII	RMII	RGMII	Description	I/O
MII_RXER	RMII_RXER	—	When asserted with RXDV, indicates the PHY detects an error in the current frame.	I
MII_TXCLK	—	—	Input clock, which provides a timing reference for TXEN, TXD[3:0], and TXER.	I
—	—	RGMII_TXC	Provides a timing reference for RGMII_TXD[3:0] and RGMII_TX_CTL. **NOTE:** This is an output signal.	O
MII_TXD[3:0]	RMII_TXD[1:0]	RGMII_TXD[3:0]	Serial output Ethernet data. Only valid during TXEN assertion.	O
MII_TXEN	RMII_TXEN	RGMII_TX_CTL	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TXCLK following the final nibble of the frame. In RGMII mode, contains TXEN on the rising edge of RGMII_TXC, and a logical derivative of TXEN and TXERR (TXEN XOR TXERR) on the falling edge of RGMII_TXC.	O
MII_TXER	—	—	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	O
—	RMII_REF_CLK	—	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	I
—	—	RGMII_REF_CLK	In RGMII mode, this signal provide 125Mhz external reference clock input.	I
1588_TMR*n*	1588_TMR*n*	1588_TMR*n*	Capture/Compare block input/output event bus.	I/O

Table continues on the next page...

MII	RMII	RGMII	Description	I/O
			When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCR*n* register for inspection by software. When configured for compare, the corresponding signal 1588_TMR*n* is asserted for 1 to 32 cycles when the timer reaches the compare value programmed in ENET_TCCR*n*. An interrupt can be triggered if ENET_TCSR*n*[TIE] is set. A DMA request can be triggered if ENET_TCSR*n*[TDRE] is set.	
ENET_1588_CLKIN	ENET_1588_CLKIN	—	Alternate IEEE 1588 Ethernet clock input; Clock period should be an integer number of nanoseconds	I

61.5 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 61-4. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

ENET memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Interrupt Event Register (ENET_EIR)	32	w1c	0000_0000h	61.5.1/3153
8	Interrupt Mask Register (ENET_EIMR)	32	R/W	0000_0000h	61.5.2/3156
10	Receive Descriptor Active Register (ENET_RDAR)	32	R/W	0000_0000h	61.5.3/3159
14	Transmit Descriptor Active Register (ENET_TDAR)	32	R/W	0000_0000h	61.5.4/3159
24	Ethernet Control Register (ENET_ECR)	32	R/W	F000_0000h	61.5.5/3160
40	MII Management Frame Register (ENET_MMFR)	32	R/W	0000_0000h	61.5.6/3162
44	MII Speed Control Register (ENET_MSCR)	32	R/W	0000_0000h	61.5.7/3163
64	MIB Control Register (ENET_MIBC)	32	R/W	C000_0000h	61.5.8/3165
84	Receive Control Register (ENET_RCR)	32	R/W	05EE_0001h	61.5.9/3166
C4	Transmit Control Register (ENET_TCR)	32	R/W	0000_0000h	61.5.10/ 3169
E4	Physical Address Lower Register (ENET_PALR)	32	R/W	0000_0000h	61.5.11/ 3171
E8	Physical Address Upper Register (ENET_PAUR)	32	R/W	0000_8808h	61.5.12/ 3171
EC	Opcode/Pause Duration Register (ENET_OPD)	32	R/W	0001_0000h	61.5.13/ 3172
F0	Transmit Interrupt Coalescing Register (ENET_TXIC)	32	R/W	0000_0000h	61.5.14/ 3172
100	Receive Interrupt Coalescing Register (ENET_RXIC)	32	R/W	0000_0000h	61.5.15/ 3173
118	Descriptor Individual Upper Address Register (ENET_IAUR)	32	R/W	0000_0000h	61.5.16/ 3174
11C	Descriptor Individual Lower Address Register (ENET_IALR)	32	R/W	0000_0000h	61.5.17/ 3175
120	Descriptor Group Upper Address Register (ENET_GAUR)	32	R/W	0000_0000h	61.5.18/ 3175
124	Descriptor Group Lower Address Register (ENET_GALR)	32	R/W	0000_0000h	61.5.19/ 3176
144	Transmit FIFO Watermark Register (ENET_TFWR)	32	R/W	0000_0000h	61.5.20/ 3176
180	Receive Descriptor Ring Start Register (ENET_RDSR)	32	R/W	0000_0000h	61.5.21/ 3177
184	Transmit Buffer Descriptor Ring Start Register (ENET_TDSR)	32	R/W	0000_0000h	61.5.22/ 3178
188	Maximum Receive Buffer Size Register (ENET_MRBR)	32	R/W	0000_0000h	61.5.23/ 3179
190	Receive FIFO Section Full Threshold (ENET_RSFL)	32	R/W	0000_0000h	61.5.24/ 3180
194	Receive FIFO Section Empty Threshold (ENET_RSEM)	32	R/W	0000_0000h	61.5.25/ 3180
198	Receive FIFO Almost Empty Threshold (ENET_RAEM)	32	R/W	0000_0004h	61.5.26/ 3181

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
19C	Receive FIFO Almost Full Threshold (ENET_RAFL)	32	R/W	0000_0004h	61.5.27/ 3181
1A0	Transmit FIFO Section Empty Threshold (ENET_TSEM)	32	R/W	0000_0000h	61.5.28/ 3182
1A4	Transmit FIFO Almost Empty Threshold (ENET_TAEM)	32	R/W	0000_0004h	61.5.29/ 3182
1A8	Transmit FIFO Almost Full Threshold (ENET_TAFL)	32	R/W	0000_0008h	61.5.30/ 3183
1AC	Transmit Inter-Packet Gap (ENET_TIPG)	32	R/W	0000_000Ch	61.5.31/ 3183
1B0	Frame Truncation Length (ENET_FTRL)	32	R/W	0000_07FFh	61.5.32/ 3184
1C0	Transmit Accelerator Function Configuration (ENET_TACC)	32	R/W	0000_0000h	61.5.33/ 3184
1C4	Receive Accelerator Function Configuration (ENET_RACC)	32	R/W	0000_0000h	61.5.34/ 3185
200	Reserved Statistic Register (ENET_RMON_T_DROP)	32	R	0000_0000h	61.5.35/ 3186
204	Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)	32	R	0000_0000h	61.5.36/ 3187
208	Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)	32	R	0000_0000h	61.5.37/ 3187
20C	Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)	32	R	0000_0000h	61.5.38/ 3188
210	Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)	32	R	0000_0000h	61.5.39/ 3188
214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)	32	R	0000_0000h	61.5.40/ 3189
218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)	32	R	0000_0000h	61.5.41/ 3189
21C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)	32	R	0000_0000h	61.5.42/ 3190
220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)	32	R	0000_0000h	61.5.43/ 3190
224	Tx Collision Count Statistic Register (ENET_RMON_T_COL)	32	R	0000_0000h	61.5.44/ 3191
228	Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)	32	R	0000_0000h	61.5.45/ 3191
22C	Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)	32	R	0000_0000h	61.5.46/ 3192
230	Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)	32	R	0000_0000h	61.5.47/ 3192
234	Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)	32	R	0000_0000h	61.5.48/ 3193

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
238	Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)	32	R	0000_0000h	61.5.49/ 3193
23C	Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)	32	R	0000_0000h	61.5.50/ 3194
240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)	32	R	0000_0000h	61.5.51/ 3194
244	Tx Octets Statistic Register (ENET_RMON_T_OCTETS)	32	R	0000_0000h	61.5.52/ 3195
248	Reserved Statistic Register (ENET_IEEE_T_DROP)	32	R	0000_0000h	61.5.53/ 3195
24C	Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)	32	R	0000_0000h	61.5.54/ 3195
250	Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)	32	R	0000_0000h	61.5.55/ 3196
254	Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)	32	R	0000_0000h	61.5.56/ 3196
258	Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)	32	R	0000_0000h	61.5.57/ 3197
25C	Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)	32	R	0000_0000h	61.5.58/ 3197
260	Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)	32	R	0000_0000h	61.5.59/ 3198
264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)	32	R	0000_0000h	61.5.60/ 3198
268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)	32	R	0000_0000h	61.5.61/ 3199
26C	Reserved Statistic Register (ENET_IEEE_T_SQE)	32	R (reads 0)	0000_0000h	61.5.62/ 3199
270	Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)	32	R	0000_0000h	61.5.63/ 3200
274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)	32	R	0000_0000h	61.5.64/ 3200
284	Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)	32	R	0000_0000h	61.5.65/ 3201
288	Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)	32	R	0000_0000h	61.5.66/ 3201
28C	Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)	32	R	0000_0000h	61.5.67/ 3202
290	Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)	32	R	0000_0000h	61.5.68/ 3202
294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)	32	R	0000_0000h	61.5.69/ 3203
298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)	32	R	0000_0000h	61.5.70/ 3203

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)	32	R	0000_0000h	61.5.71/ 3204
2A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)	32	R	0000_0000h	61.5.72/ 3204
2A4	Reserved Statistic Register (ENET_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	61.5.73/ 3204
2A8	Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)	32	R	0000_0000h	61.5.74/ 3205
2AC	Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)	32	R	0000_0000h	61.5.75/ 3205
2B0	Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)	32	R	0000_0000h	61.5.76/ 3206
2B4	Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)	32	R	0000_0000h	61.5.77/ 3206
2B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)	32	R	0000_0000h	61.5.78/ 3207
2BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)	32	R	0000_0000h	61.5.79/ 3207
2C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)	32	R	0000_0000h	61.5.80/ 3208
2C4	Rx Octets Statistic Register (ENET_RMON_R_OCTETS)	32	R	0000_0000h	61.5.81/ 3208
2C8	Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)	32	R	0000_0000h	61.5.82/ 3209
2CC	Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)	32	R	0000_0000h	61.5.83/ 3209
2D0	Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)	32	R	0000_0000h	61.5.84/ 3210
2D4	Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)	32	R	0000_0000h	61.5.85/ 3210
2D8	Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)	32	R	0000_0000h	61.5.86/ 3211
2DC	Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)	32	R	0000_0000h	61.5.87/ 3211
2E0	Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)	32	R	0000_0000h	61.5.88/ 3212
400	Adjustable Timer Control Register (ENET_ATCR)	32	R/W	0000_0000h	61.5.89/ 3212
404	Timer Value Register (ENET_ATVR)	32	R/W	0000_0000h	61.5.90/ 3214
408	Timer Offset Register (ENET_ATOFF)	32	R/W	0000_0000h	61.5.91/ 3214
40C	Timer Period Register (ENET_ATPER)	32	R/W	3B9A_CA00h	61.5.92/ 3215

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
410	Timer Correction Register (ENET_ATCOR)	32	R/W	0000_0000h	61.5.93/ 3215
414	Time-Stamping Clock Period Register (ENET_ATINC)	32	R/W	0000_0000h	61.5.94/ 3216
418	Timestamp of Last Transmitted Frame (ENET_ATSTMP)	32	R	0000_0000h	61.5.95/ 3216
604	Timer Global Status Register (ENET_TGSR)	32	R/W	0000_0000h	61.5.96/ 3217
608	Timer Control Status Register (ENET_TCSR0)	32	R/W	0000_0000h	61.5.97/ 3218
60C	Timer Compare Capture Register (ENET_TCCR0)	32	R/W	0000_0000h	61.5.98/ 3219
610	Timer Control Status Register (ENET_TCSR1)	32	R/W	0000_0000h	61.5.97/ 3218
614	Timer Compare Capture Register (ENET_TCCR1)	32	R/W	0000_0000h	61.5.98/ 3219
618	Timer Control Status Register (ENET_TCSR2)	32	R/W	0000_0000h	61.5.97/ 3218
61C	Timer Compare Capture Register (ENET_TCCR2)	32	R/W	0000_0000h	61.5.98/ 3219
620	Timer Control Status Register (ENET_TCSR3)	32	R/W	0000_0000h	61.5.97/ 3218
624	Timer Compare Capture Register (ENET_TCCR3)	32	R/W	0000_0000h	61.5.98/ 3219

61.5.1 Interrupt Event Register (ENET_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode, ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER															
W	w1c	0		0	0			0	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
4 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
7 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
8 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
9 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.
10 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
12 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13 PLR	Payload Receive Error Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
14 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
15 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
16 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
17–18 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
19 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24–31 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

61.5.2 Interrupt Mask Register (ENET_EIMR)

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER															
W		0		0		0		0				0				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIMR field descriptions

Field	Description
0 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
1 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
2 BABT	BABT Interrupt Mask Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
3 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
	corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
4 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
5 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
6 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
7 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
8 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
9 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
10 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

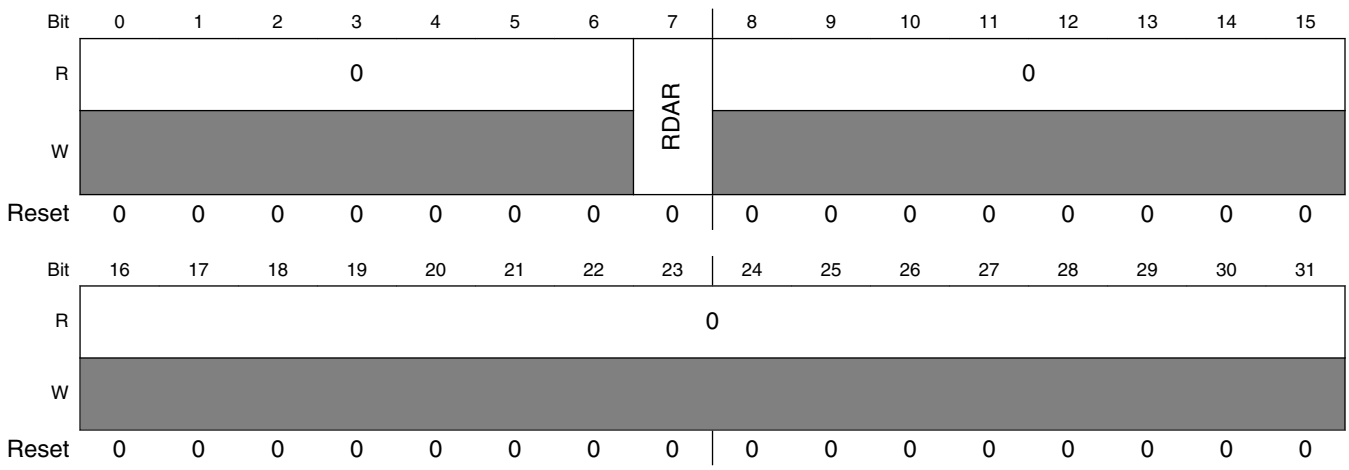
ENET_EIMR field descriptions (continued)

Field	Description
11 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
12 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
13 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
14 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
15 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
16 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
17–18 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
19 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24–31 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

61.5.3 Receive Descriptor Active Register (ENET_RDAR)

RDAR is a command register, written by the user, to indicate that the receive descriptor ring has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 0h base + 10h offset = 10h



ENET_RDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.4 Transmit Descriptor Active Register (ENET_TDAR)

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							TDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.5 Ethernet Control Register (ENET_ECR)

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved														Reserved	
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				Reserved	Reserved	Reserved	DBSWP	STOPEN	DBGEN	SPEED	EN1588	SLEEP	MAGICEN	ETHEREN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ECR field descriptions

Field	Description
0–13 Reserved	This field is reserved. Always write 11110000000000b to this field.
14–19 Reserved	This field is reserved. Always write 0 to this field.
20 Reserved	This field is reserved. Always write 0 to this field.
21 Reserved	This field is reserved. Always write 0 to this field.
22 Reserved	This field is reserved. Always write 0 to this field.
23 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field resets to 0 and must not be changed. 0 The buffer descriptor bytes are not swapped to support big-endian devices. 1 The buffer descriptor bytes are swapped to support little-endian devices.
24 STOPEN	STOPEN Signal Control Controls device behavior in doze mode. In doze mode, if this field is set then all the clocks of the ENET assembly are disabled, except the RMII /MII clock. Doze mode is similar to a conditional stop mode entry for the ENET assembly depending on ECR[STOPEN]. **NOTE:** If module clocks are gated in this mode, the module can still wake the system after receiving a magic packet in stop mode. MAGICEN must be set prior to entering sleep/stop mode.
25 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0 MAC continues operation in debug mode. 1 MAC enters hardware freeze mode when the processor is in debug mode.
26 SPEED	Selects between 10/100-Mbit/s and 1000-Mbit/s modes of operation. 0 10/100-Mbit/s mode 1 1000-Mbit/s mode (Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.)
27 EN1588	EN1588 Enable Enables enhanced functionality of the MAC. 0 Legacy FEC buffer descriptors and functions enabled. 1 Enhanced frame time-stamping functions enabled.
28 SLEEP	Sleep Mode Enable 0 Normal operating mode. 1 Sleep mode.
29 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
	NOTE: MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. 0 Magic detection logic disabled. 1 The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.
30 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0 Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1 MAC is enabled, and reception and transmission are possible.
31 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

61.5.6 MII Management Frame Register (ENET_MMFR)

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MMFR field descriptions

Field	Description
0–1 ST	Start Of Frame Delimiter See Table 61-42 (Clause 22) or Table 61-44 (Clause 45) for correct value.
2–3 OP	Operation Code See Table 61-42 (Clause 22) or Table 61-44 (Clause 45) for correct value.
4–8 PA	PHY Address See Table 61-42 (Clause 22) or Table 61-44 (Clause 45) for correct value.
9–13 RA	Register Address See Table 61-42 (Clause 22) or Table 61-44 (Clause 45) for correct value.
14–15 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

61.5.7 MII Speed Control Register (ENET_MSCR)

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (i.e., IPS bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 61-5. Programming Examples for MSCR

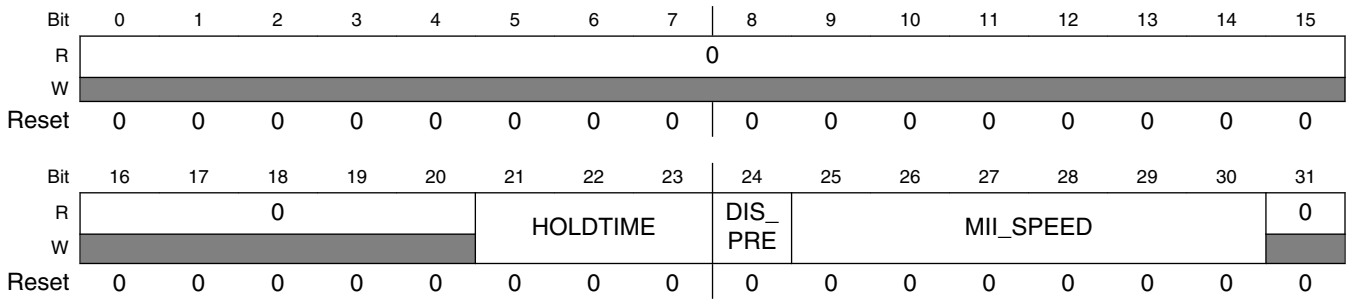
Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz

Table continues on the next page...

Table 61-5. Programming Examples for MSCR (continued)

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz
66 MHz	0xD	2.36 MHz

Address: 0h base + 44h offset = 44h



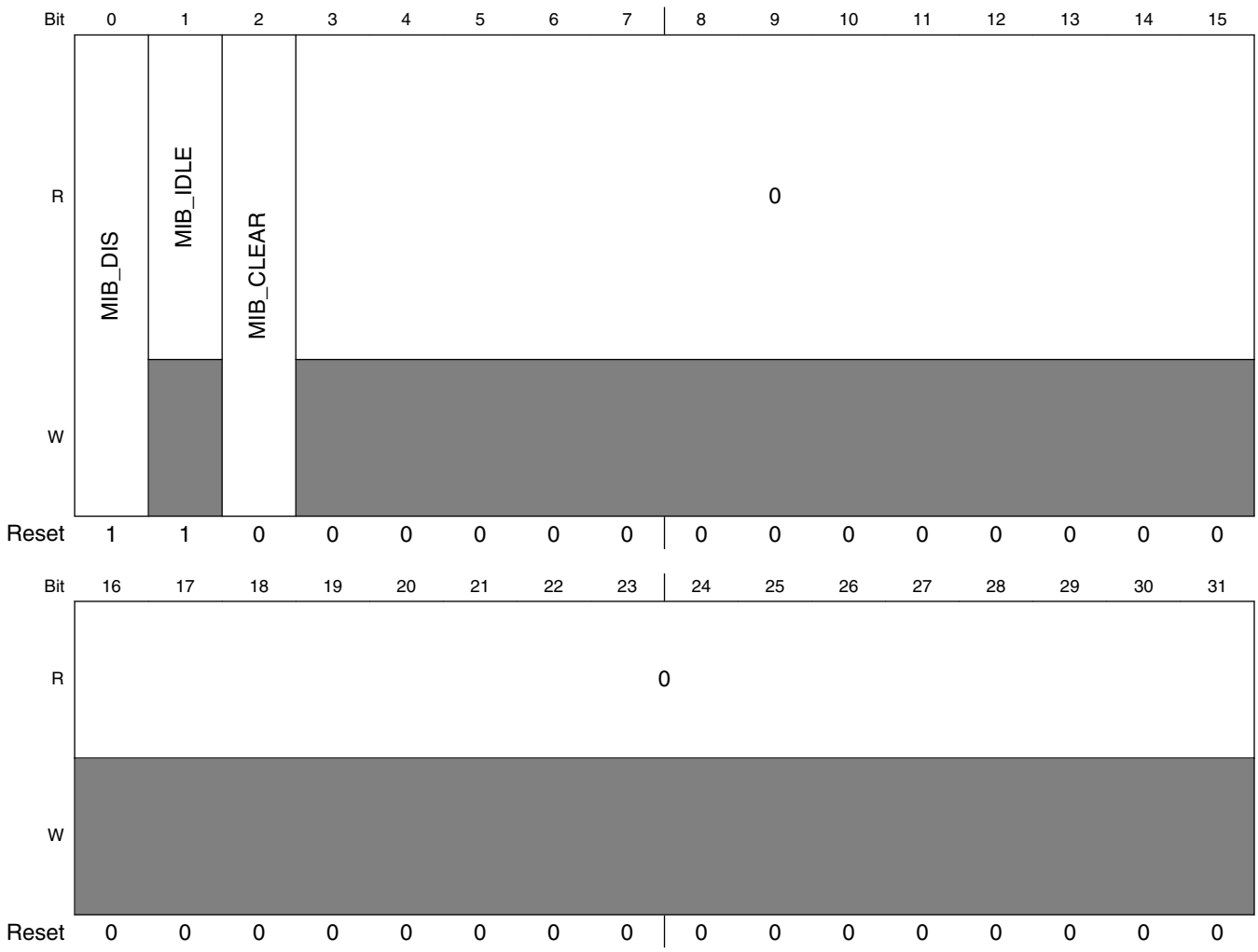
ENET_MSCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000 1 internal module clock cycle 001 2 internal module clock cycles 010 3 internal module clock cycles 111 8 internal module clock cycles
24 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0 Preamble enabled. 1 Preamble (32 ones) is not prepended to the MII management frame.
25–30 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.8 MIB Control Register (ENET_MIBC)

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

Address: 0h base + 64h offset = 64h



ENET_MIBC field descriptions

Field	Description
0 MIB_DIS	Disable MIB Logic If this control field is set, 0 MIB logic is enabled. 1 MIB logic is disabled. The MIB logic halts and does not update any MIB counters.

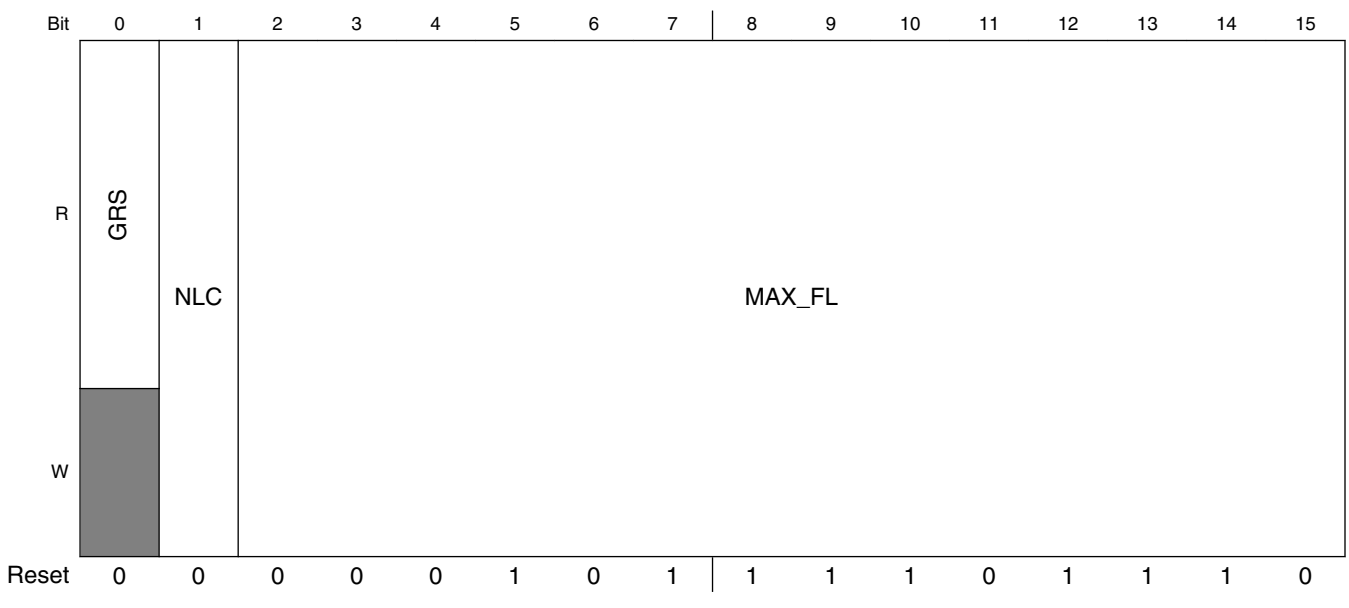
Table continues on the next page...

ENET_MIBC field descriptions (continued)

Field	Description
1 MIB_IDLE	MIB Idle 0 The MIB block is updating MIB counters. 1 The MIB block is not currently updating any MIB counters.
2 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0 See note above. 1 All statistics counters are reset to 0.
3–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.9 Receive Control Register (ENET_RCR)

Address: 0h base + 84h offset = 84h



ENET_RCR field descriptions

Field	Description
0 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped.
1 NLC	Payload Length Check Disable Enables/disables a payload length check. 0 The payload length check is disabled. 1 The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLC] field.
2–15 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16 CFEN	MAC Control Frame Enable Enables/disables the MAC control frame. 0 MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1 MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
17 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0 The CRC field of received frames is transmitted to the user application. 1 The CRC field is stripped from the frame.
18 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0 Pause frames are terminated and discarded in the MAC. 1 Pause frames are forwarded to the user application.
19 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0 No padding is removed on receive by the MAC. 1 Padding is removed from received frames.
20–21 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
22 RMII_10T	Enables 10-Mbit/s mode of the RMII or RGMII . 0 100-Mbit/s operation. 1 10-Mbit/s operation.

Table continues on the next page...

ENET_RCR field descriptions (continued)

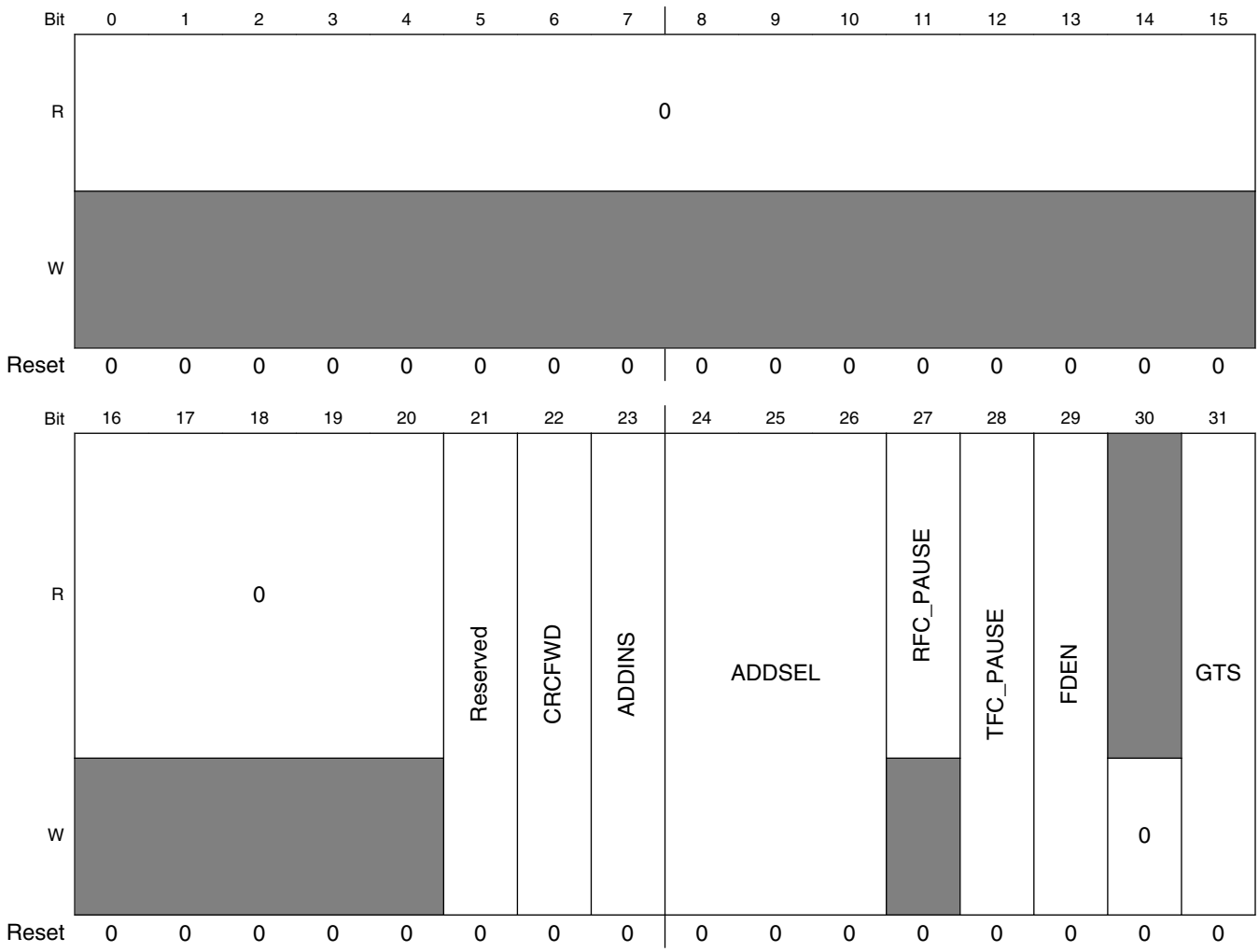
Field	Description
23 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation , when ECR[SPEED] is cleared . **NOTE:** Do not set both RCR[RGMII_EN] and RCR[RMII_MODE]. 0 MAC configured for MII mode. 1 MAC configured for RMII operation.
24 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
25 RGMII_EN	RGMII Mode Enable **NOTE:** Do not set both RCR[RGMII_EN] and RCR[RMII_MODE]. 0 MAC configured for non-RGMII operation 1 MAC configured for RGMII operation. If ECR[SPEED] is set, the MAC is in RGMII 1000-Mbit/s[1] mode. If ECR[SPEED] is cleared, the MAC is in RGMII 10/100-Mbit/s mode.
26 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor.
28 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0 Disabled. 1 Enabled.
29 MII_MODE	Media Independent Interface Mode This field must always be set. 0 Reserved. 1 MII or RMII mode, as indicated by the RMII_MODE field.
30 DRT	Disable Receive On Transmit 0 Receive path operates independently of transmit (i.e., full-duplex mode). Can also be used to monitor transmit activity in half-duplex mode. 1 Disable reception of frames while transmitting. (Normally used for half-duplex mode.)
31 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0 Loopback disabled. 1 Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

1. Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.

61.5.10 Transmit Control Register (ENET_TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

Address: 0h base + C4h offset = C4h



ENET_TCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This field is read/write and must be set to 0.
22 CRCFWD	Forward Frame From Application With CRC

Table continues on the next page...

ENET_TCR field descriptions (continued)

Field	Description
	0 TxBD[TC] controls whether the frame has a CRC from the application. 1 The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.
23 ADDINS	Set MAC Address On Transmit 0 The source MAC address is not modified by the MAC. 1 The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
24–26 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000 Node MAC address programmed on PADDR1/2 registers. 100 Reserved. 101 Reserved. 110 Reserved.
27 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0 No PAUSE frame transmitted. 1 The MAC stops transmission of data frames after the current transmission is complete.
29 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared.
30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

61.5.11 Physical Address Lower Register (ENET_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PALR field descriptions

Field	Description
0–31 PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

61.5.12 Physical Address Upper Register (ENET_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR2																TYPE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

ENET_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

61.5.13 Opcode/Pause Duration Register (ENET_OPD)

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
16–31 PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

61.5.14 Transmit Interrupt Coalescing Register (ENET_TXIC)

See Interrupt coalescence for more information.

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ICEN	ICCS	Reserved		ICFT								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ICTT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TXIC field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable

Table continues on the next page...

ENET_TXIC field descriptions (continued)

Field	Description
	0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be transmitted for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after transmitting a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame transmission defined by ICFT and starts again upon transmission of the next first frame. Must be greater than zero to avoid unpredictable behavior.

61.5.15 Receive Interrupt Coalescing Register (ENET_RXIC)

See Interrupt coalescence for more information.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ICEN	ICCS	Reserved		ICFT								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ICTT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RXIC field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable 0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.

Table continues on the next page...

ENET_RXIC field descriptions (continued)

Field	Description
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be received for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after receiving a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame reception defined by ICFT and starts again upon reception of the next first frame. Must be greater than zero to avoid unpredictable behavior.

61.5.16 Descriptor Individual Upper Address Register (ENET_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IAUR field descriptions

Field	Description
0–31 IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

61.5.17 Descriptor Individual Lower Address Register (ENET_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IALR field descriptions

Field	Description
0–31 IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

61.5.18 Descriptor Group Upper Address Register (ENET_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GAUR field descriptions

Field	Description
0–31 GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

61.5.19 Descriptor Group Lower Address Register (ENET_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GALR field descriptions

Field	Description
0–31 GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

61.5.20 Transmit FIFO Watermark Register (ENET_TFWR)

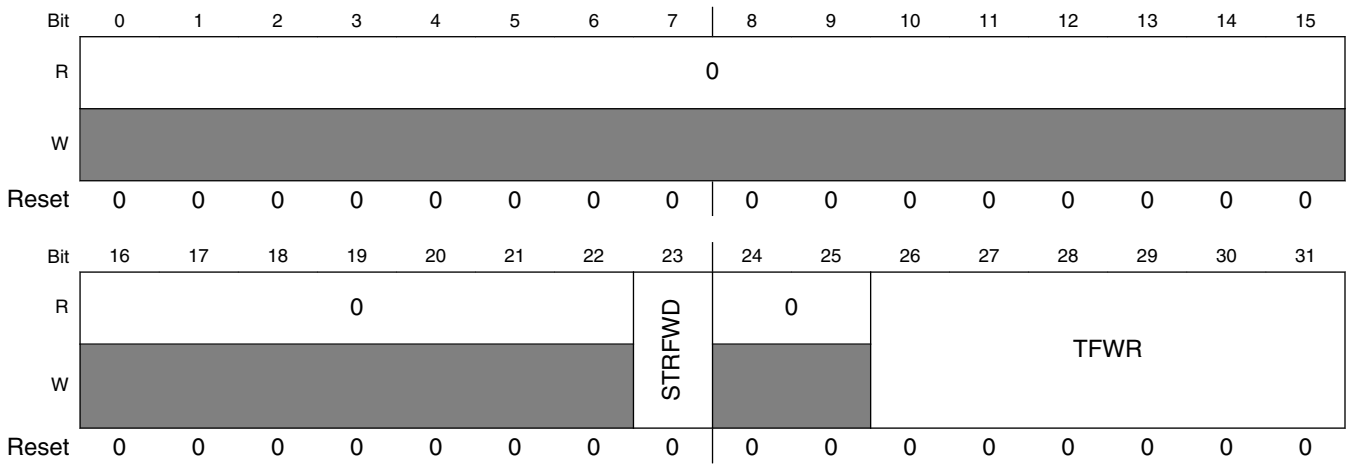
If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

Address: 0h base + 144h offset = 144h



ENET_TFWR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 STRFWD	Store And Forward Enable 0 Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1 Enabled.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000 64 bytes written. 000001 64 bytes written. 000010 128 bytes written. 000011 192 bytes written. ... ... 111111 4032 bytes written.

61.5.21 Receive Descriptor Ring Start Register (ENET_RDSR)

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.22 Transmit Buffer Descriptor Ring Start Register (ENET_TDSR)

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.

Table continues on the next page...

ENET_TDSR field descriptions (continued)

Field	Description
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.23 Maximum Receive Buffer Size Register (ENET_MRBR)

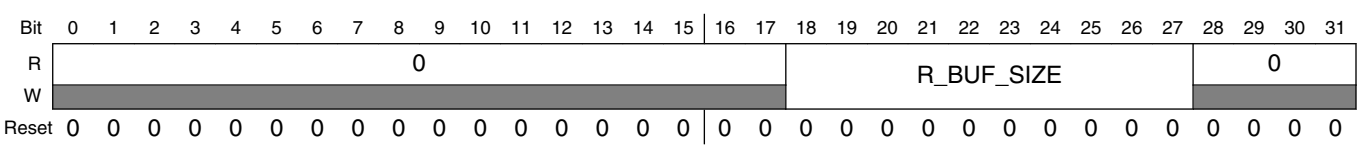
The MRBR is a user-programmable register that dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.
- To properly align the buffer, MRBR must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits of this register are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 0h base + 188h offset = 188h

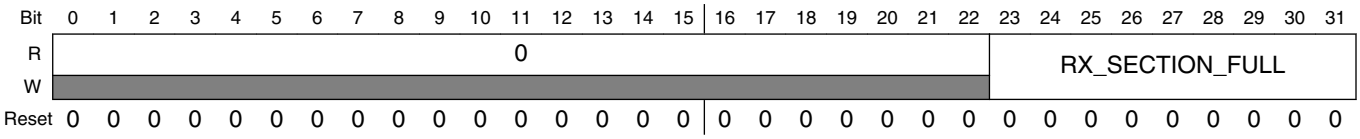


ENET_MRBR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

61.5.24 Receive FIFO Section Full Threshold (ENET_RSFL)

Address: 0h base + 190h offset = 190h

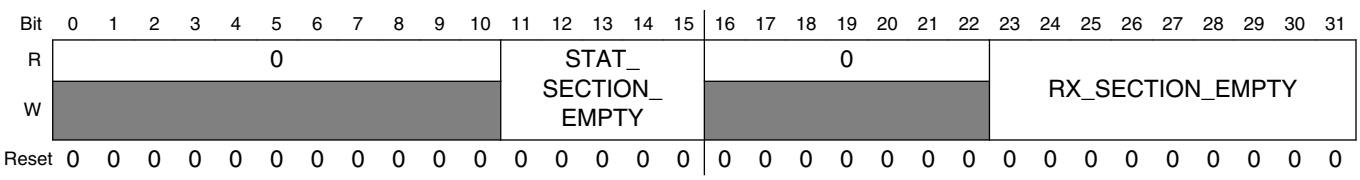


ENET_RSFL field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 RX_SECTION_FULL	Value Of Receive FIFO Section Full Threshold Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

61.5.25 Receive FIFO Section Empty Threshold (ENET_RSEM)

Address: 0h base + 194h offset = 194h



ENET_RSEM field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 STAT_ SECTION_ EMPTY	RX Status FIFO Section Empty Threshold Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 RX_SECTION_ EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued.

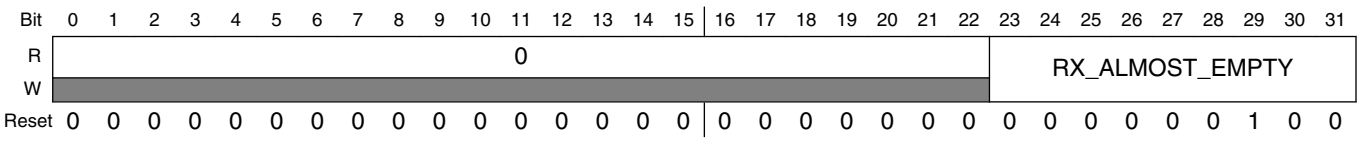
Table continues on the next page...

ENET_RSEM field descriptions (continued)

Field	Description
	A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

61.5.26 Receive FIFO Almost Empty Threshold (ENET_RAEM)

Address: 0h base + 198h offset = 198h

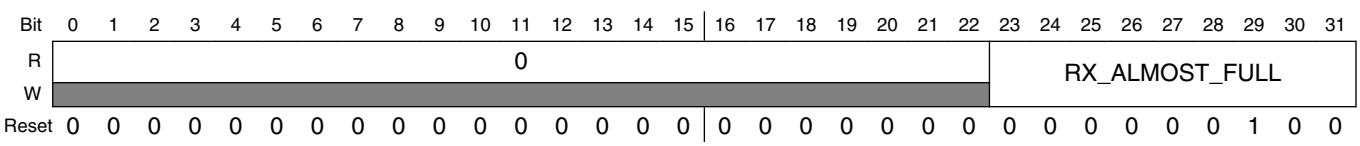


ENET_RAEM field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 RX_ALMOST_EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

61.5.27 Receive FIFO Almost Full Threshold (ENET_RAFL)

Address: 0h base + 19Ch offset = 19Ch



ENET_RAFL field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

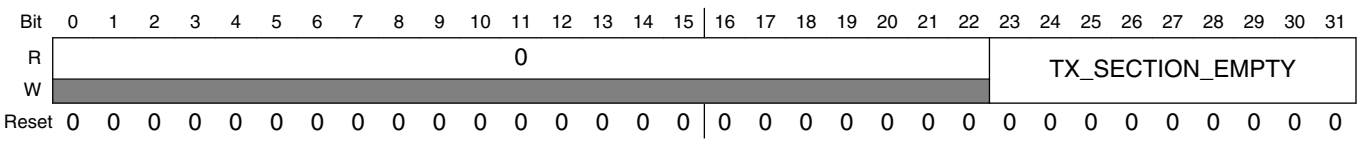
Table continues on the next page...

ENET_RAFL field descriptions (continued)

Field	Description
23–31 RX_ALMOST_FULL	Value Of The Receive FIFO Almost Full Threshold Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

61.5.28 Transmit FIFO Section Empty Threshold (ENET_TSEM)

Address: 0h base + 1A0h offset = 1A0h

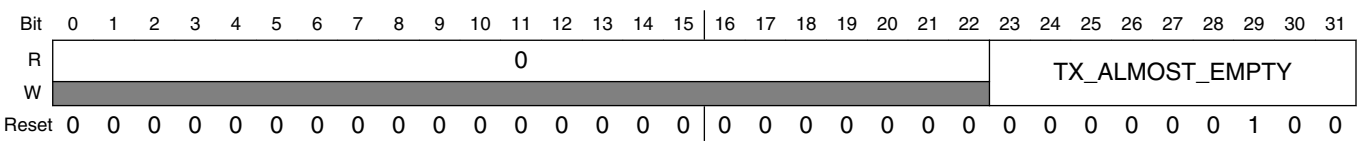


ENET_TSEM field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TX_SECTION_EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

61.5.29 Transmit FIFO Almost Empty Threshold (ENET_TAEM)

Address: 0h base + 1A4h offset = 1A4h



ENET_TAEM field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TX_ALMOST_EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold.

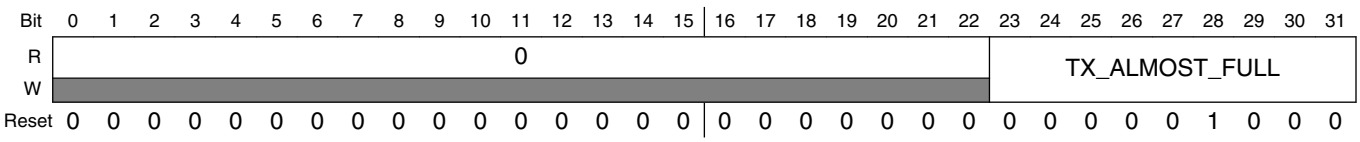
Table continues on the next page...

ENET_TAEM field descriptions (continued)

Field	Description
	When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

61.5.30 Transmit FIFO Almost Full Threshold (ENET_TAFL)

Address: 0h base + 1A8h offset = 1A8h

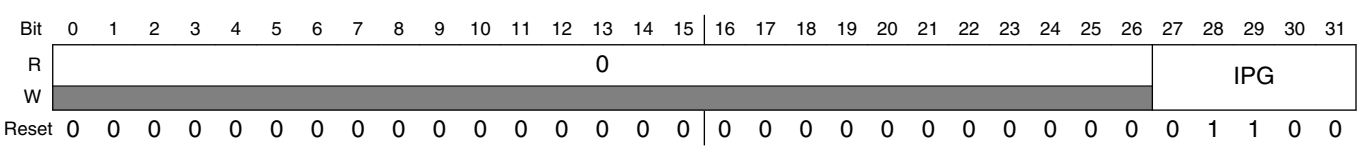


ENET_TAFL field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TX_ALMOST_FULL	Value Of The Transmit FIFO Almost Full Threshold Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

61.5.31 Transmit Inter-Packet Gap (ENET_TIPG)

Address: 0h base + 1ACh offset = 1ACh

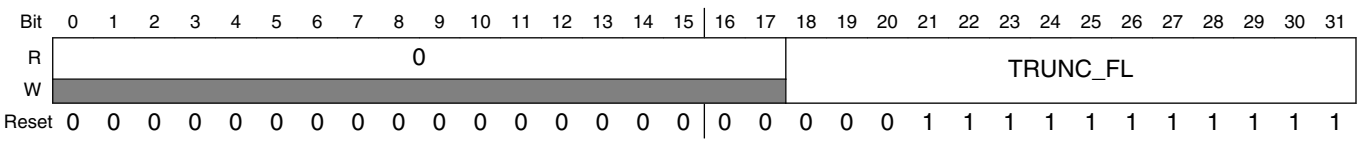


ENET_TIPG field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

61.5.32 Frame Truncation Length (ENET_FTRL)

Address: 0h base + 1B0h offset = 1B0h



ENET_FTRL field descriptions

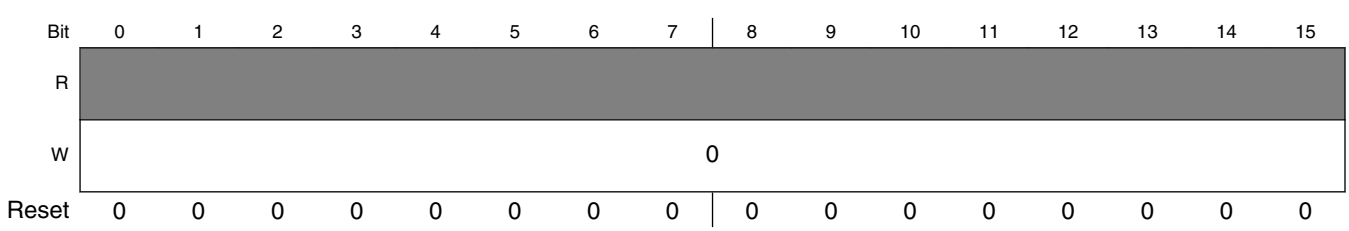
Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

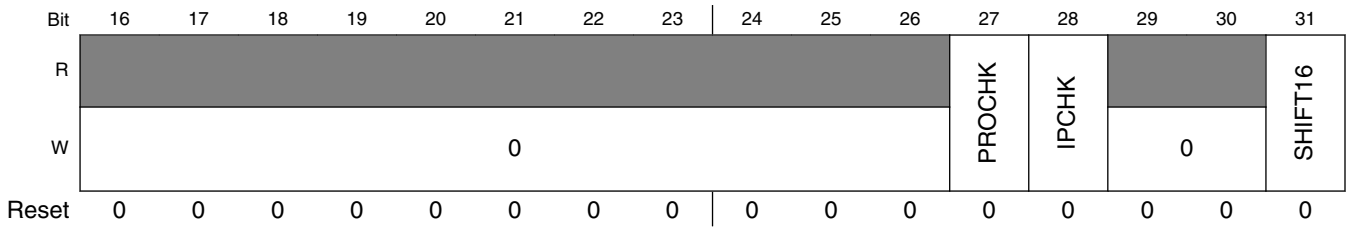
61.5.33 Transmit Accelerator Function Configuration (ENET_TACC)

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

Address: 0h base + 1C0h offset = 1C0h



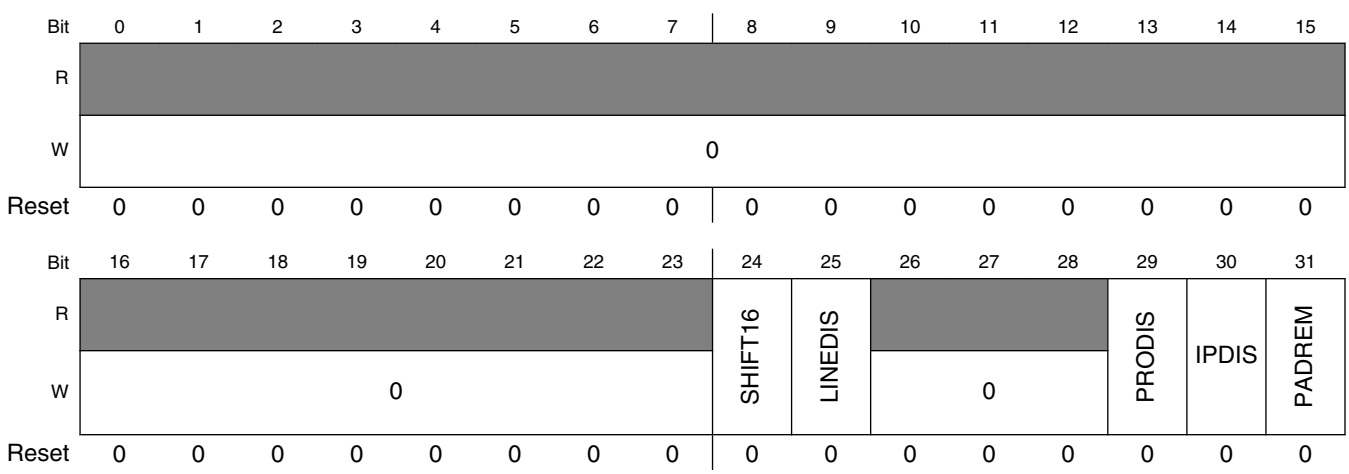


ENET_TACC field descriptions

Field	Description
0–26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 PROCHK	Enables insertion of protocol checksum. 0 Checksum not inserted. 1 If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
28 IPCHK	Enables insertion of IP header checksum. 0 Checksum is not inserted. 1 If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
29–30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 SHIFT16	TX FIFO Shift-16 0 Disabled. 1 Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

61.5.34 Receive Accelerator Function Configuration (ENET_RACC)

Address: 0h base + 1C4h offset = 1C4h

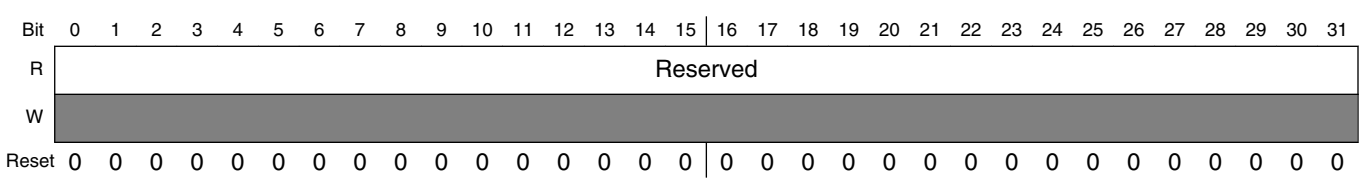


ENET_RACC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0 Disabled. 1 Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.
25 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0 Frames with errors are not discarded. 1 Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
26–28 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
29 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0 Frames with wrong checksum are not discarded. 1 If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
30 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0 Frames with wrong IPv4 header checksum are not discarded. 1 If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
31 PADREM	Enable Padding Removal For Short IP Frames 0 Padding not removed. 1 Any bytes following the IP payload section of the frame are removed from the frame.

61.5.35 Reserved Statistic Register (ENET_RMON_T_DROP)

Address: 0h base + 200h offset = 200h

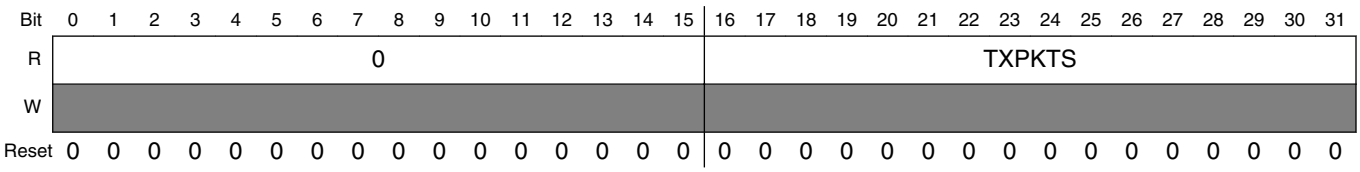


ENET_RMON_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

61.5.36 Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)

Address: 0h base + 204h offset = 204h



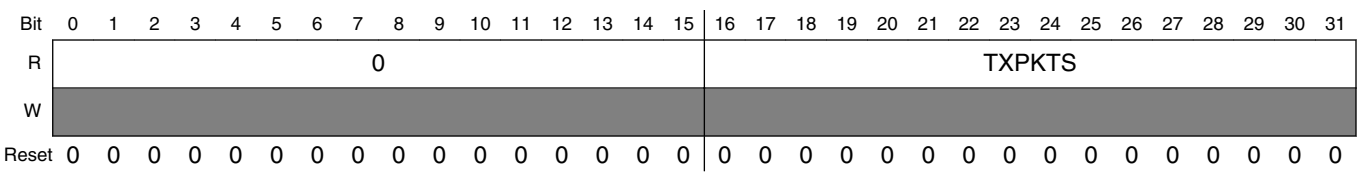
ENET_RMON_T_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packet count Transmit packet count

61.5.37 Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)

RMON Tx Broadcast Packets

Address: 0h base + 208h offset = 208h

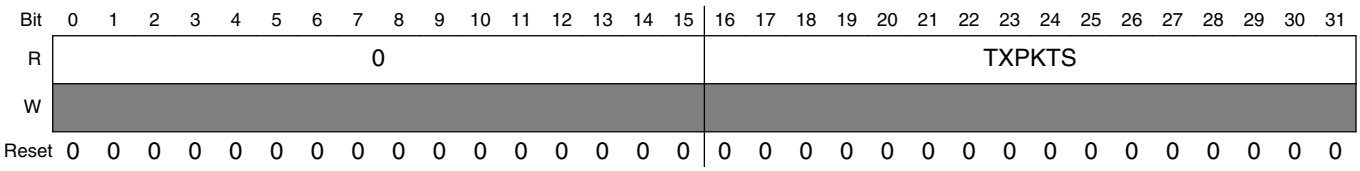


ENET_RMON_T_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Broadcast packets

61.5.38 Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)

Address: 0h base + 20Ch offset = 20Ch

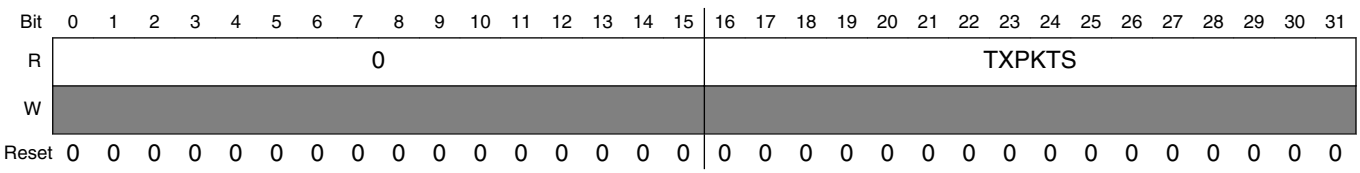


ENET_RMON_T_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Multicast packets

61.5.39 Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)

Address: 0h base + 210h offset = 210h



ENET_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packets with CRC/align error

61.5.40 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)

Address: 0h base + 214h offset = 214h

Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

ENET_RMON_T_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets less than 64 bytes with good CRC

61.5.41 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)

Address: 0h base + 218h offset = 218h

Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

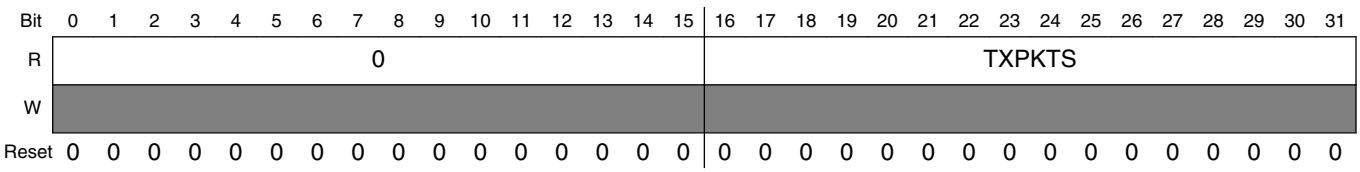
ENET_RMON_T_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

61.5.42 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)

.

Address: 0h base + 21Ch offset = 21Ch

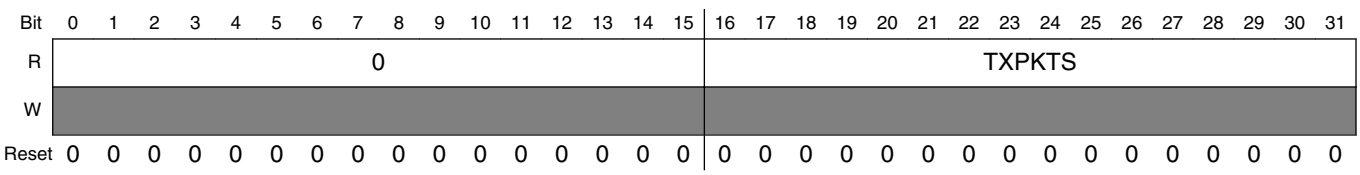


ENET_RMON_T_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of packets less than 64 bytes with bad CRC

61.5.43 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)

Address: 0h base + 220h offset = 220h



ENET_RMON_T_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

61.5.44 Tx Collision Count Statistic Register (ENET_RMON_T_COL)

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit collisions

61.5.45 Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)

.

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 64-byte transmit packets

61.5.46 Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)

Address: 0h base + 22Ch offset = 22Ch

Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

ENET_RMON_T_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 65- to 127-byte transmit packets

61.5.47 Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)

Address: 0h base + 230h offset = 230h

Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

ENET_RMON_T_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 128- to 255-byte transmit packets

61.5.48 Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 256- to 511-byte transmit packets

61.5.49 Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)

.

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 512- to 1023-byte transmit packets

61.5.50 Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)

Address: 0h base + 23Ch offset = 23Ch

Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

ENET_RMON_T_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 1024- to 2047-byte transmit packets

61.5.51 Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)

Address: 0h base + 240h offset = 240h

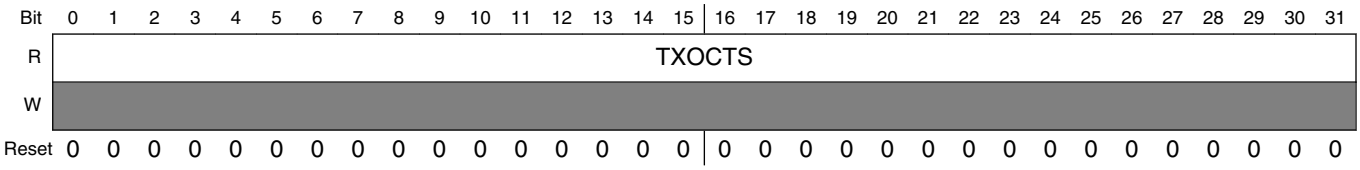
Bit	0–15	16–31
R	0	TXPKTS
W		
Reset	0	0

ENET_RMON_T_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than 2048 bytes

61.5.52 Tx Octets Statistic Register (ENET_RMON_T_OCTETS)

Address: 0h base + 244h offset = 244h

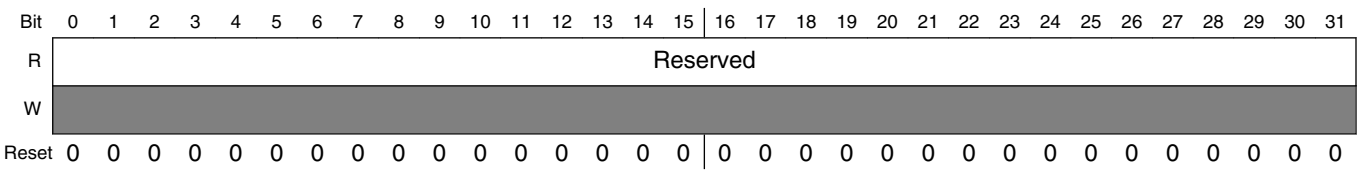


ENET_RMON_T_OCTETS field descriptions

Field	Description
0–31 TXOCTS	Number of transmit octets

61.5.53 Reserved Statistic Register (ENET_IEEE_T_DROP)

Address: 0h base + 248h offset = 248h

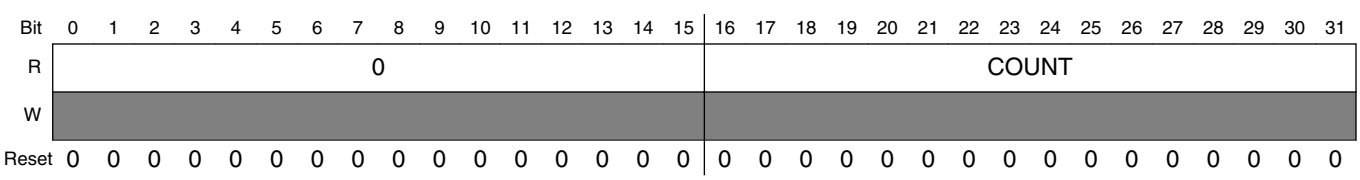


ENET_IEEE_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

61.5.54 Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)

Address: 0h base + 24Ch offset = 24Ch



ENET_IEEE_T_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

61.5.55 Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_1COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with one collision

61.5.56 Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)

Address: 0h base + 254h offset = 254h

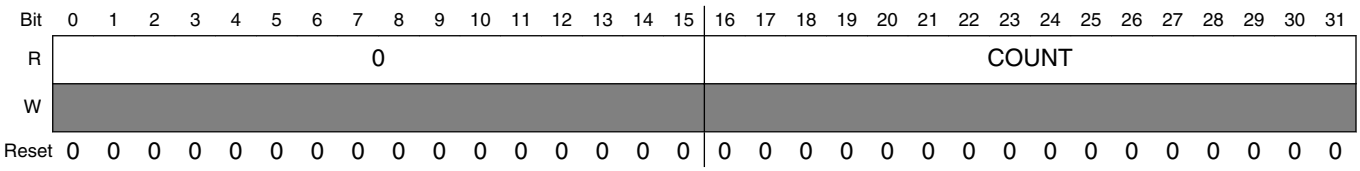
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with multiple collisions

61.5.57 Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)

Address: 0h base + 258h offset = 258h

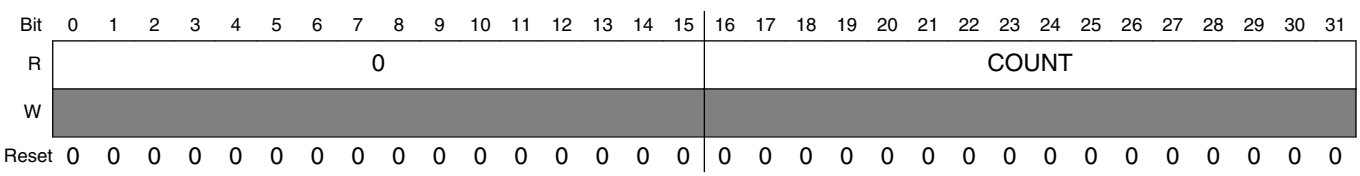


ENET_IEEE_T_DEF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with deferral delay

61.5.58 Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)

Address: 0h base + 25Ch offset = 25Ch

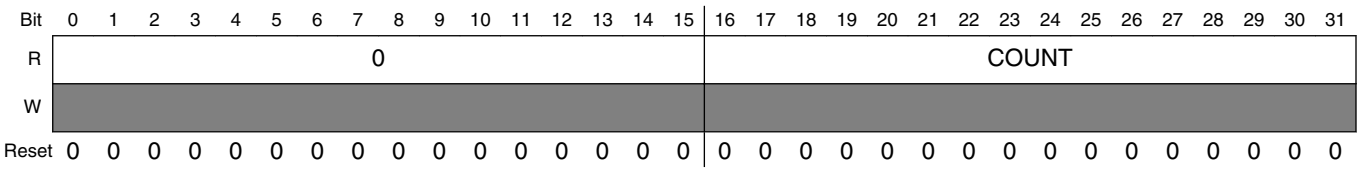


ENET_IEEE_T_LCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with late collision

61.5.59 Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)

Address: 0h base + 260h offset = 260h

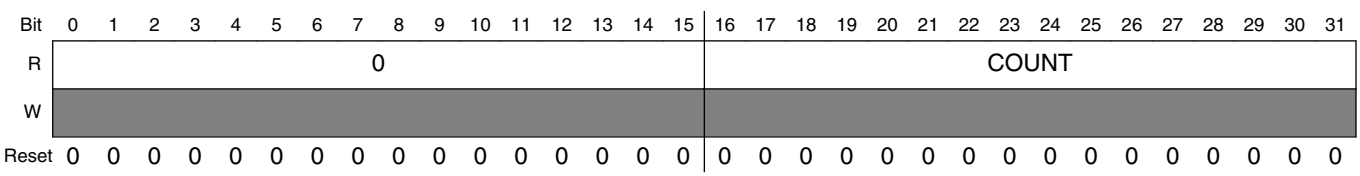


ENET_IEEE_T_EXCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with excessive collisions

61.5.60 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)

Address: 0h base + 264h offset = 264h

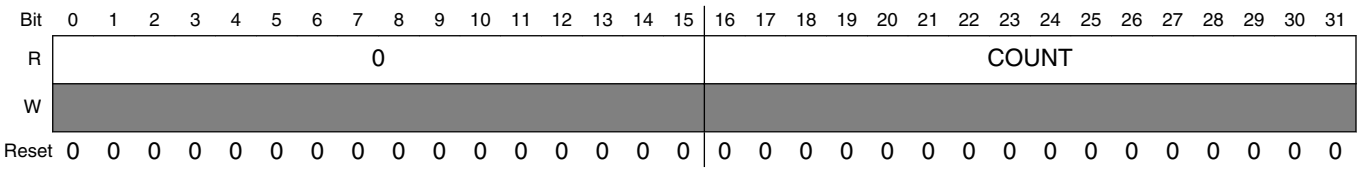


ENET_IEEE_T_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with transmit FIFO underrun

61.5.61 Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)

Address: 0h base + 268h offset = 268h

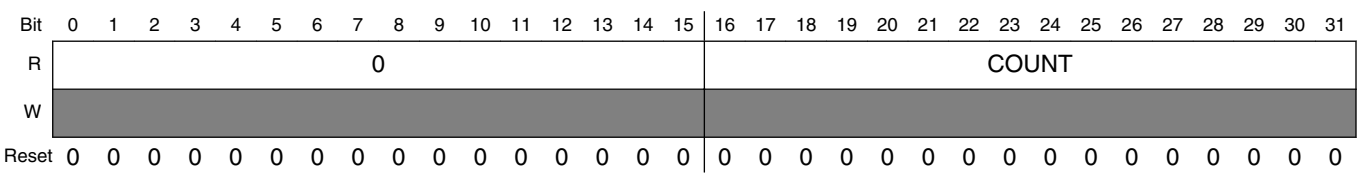


ENET_IEEE_T_CSERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with carrier sense error

61.5.62 Reserved Statistic Register (ENET_IEEE_T_SQE)

Address: 0h base + 26Ch offset = 26Ch

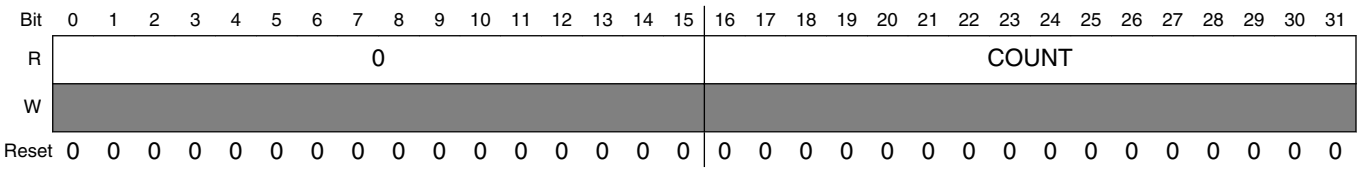


ENET_IEEE_T_SQE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	This read-only field is reserved and always has the value 0. **NOTE:** Counter not implemented as no SQE information is available.

61.5.63 Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)

Address: 0h base + 270h offset = 270h

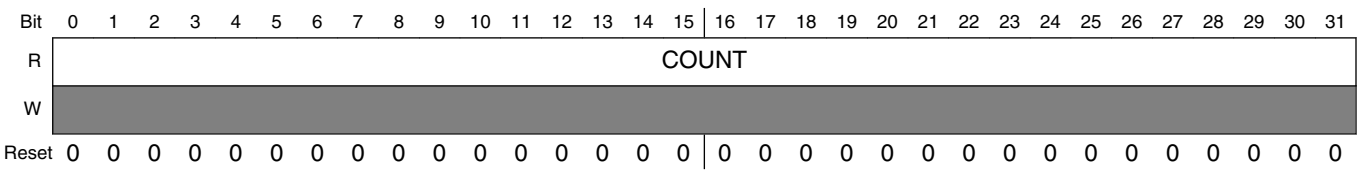


ENET_IEEE_T_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames transmitted

61.5.64 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)

Address: 0h base + 274h offset = 274h

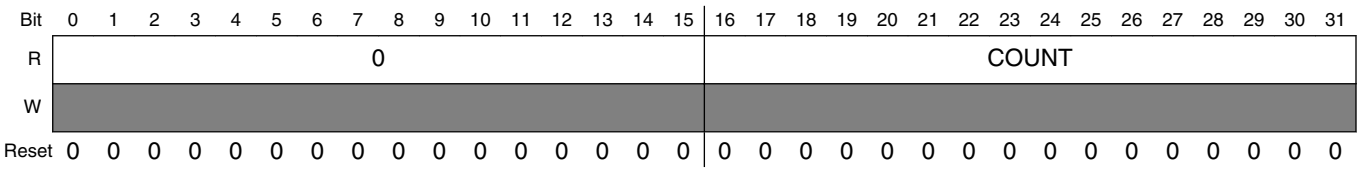


ENET_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Octet count for frames transmitted without error **NOTE** Counts total octets (includes header and FCS fields).

61.5.65 Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)

Address: 0h base + 284h offset = 284h

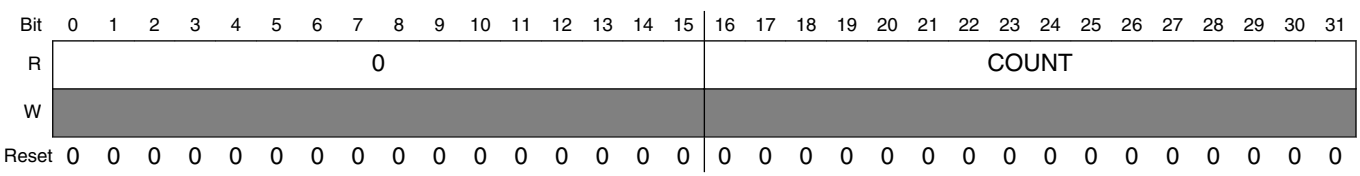


ENET_RMON_R_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of packets received

61.5.66 Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)

Address: 0h base + 288h offset = 288h

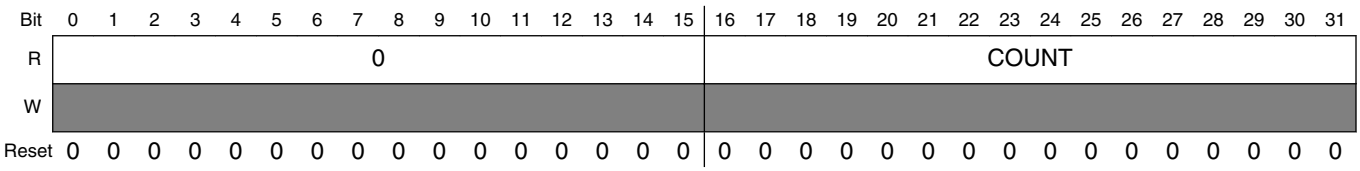


ENET_RMON_R_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive broadcast packets

61.5.67 Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)

Address: 0h base + 28Ch offset = 28Ch

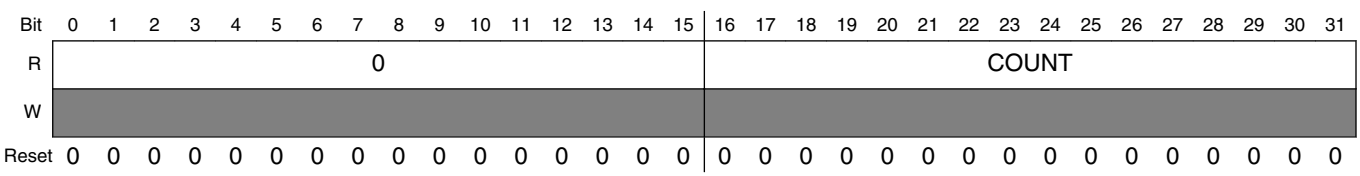


ENET_RMON_R_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive multicast packets

61.5.68 Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)

Address: 0h base + 290h offset = 290h

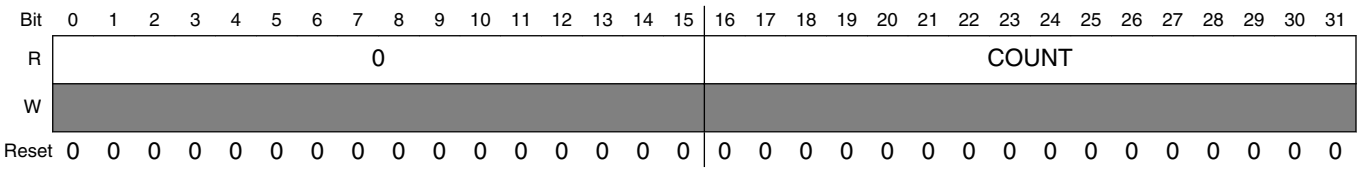


ENET_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with CRC or align error

61.5.69 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)

Address: 0h base + 294h offset = 294h

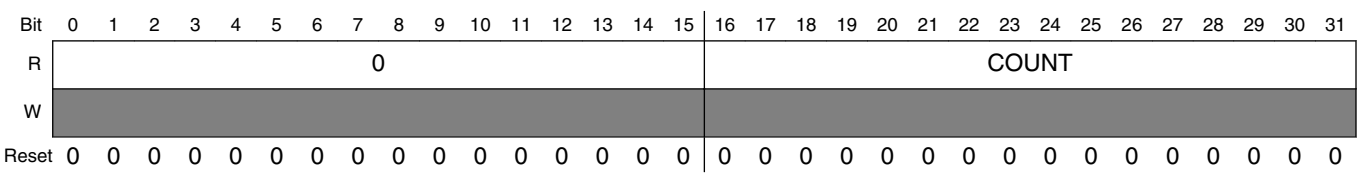


ENET_RMON_R_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and good CRC

61.5.70 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)

Address: 0h base + 298h offset = 298h



ENET_RMON_R_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and good CRC

61.5.71 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)

Address: 0h base + 29Ch offset = 29Ch

Bit	0–15	16–31
R	0	COUNT
W		
Reset	0	0

ENET_RMON_R_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and bad CRC

61.5.72 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)

Address: 0h base + 2A0h offset = 2A0h

Bit	0–15	16–31
R	0	COUNT
W		
Reset	0	0

ENET_RMON_R_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and bad CRC

61.5.73 Reserved Statistic Register (ENET_RMON_R_RESVD_0)

Address: 0h base + 2A4h offset = 2A4h

Bit	0–15	16–31
R	0	0
W		
Reset	0	0

ENET_RMON_R_RESVD_0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

61.5.74 Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)

Address: 0h base + 2A8h offset = 2A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 64-byte receive packets

61.5.75 Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 65- to 127-byte recieve packets

61.5.76 Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 128- to 255-byte recieve packets

61.5.77 Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 256- to 511-byte recieve packets

61.5.78 Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 512- to 1023-byte recieve packets

61.5.79 Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)

Address: 0h base + 2BCh offset = 2BCh

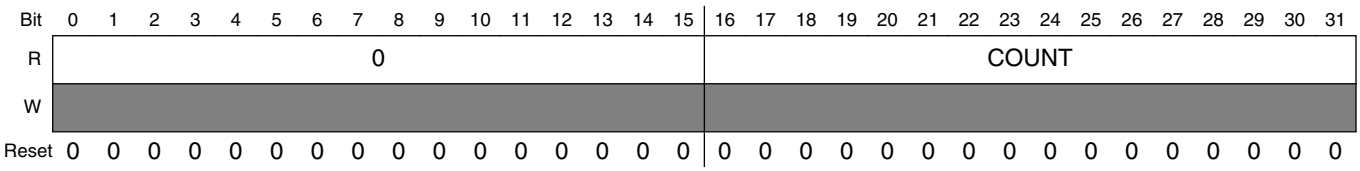
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 1024- to 2047-byte recieve packets

61.5.80 Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)

Address: 0h base + 2C0h offset = 2C0h

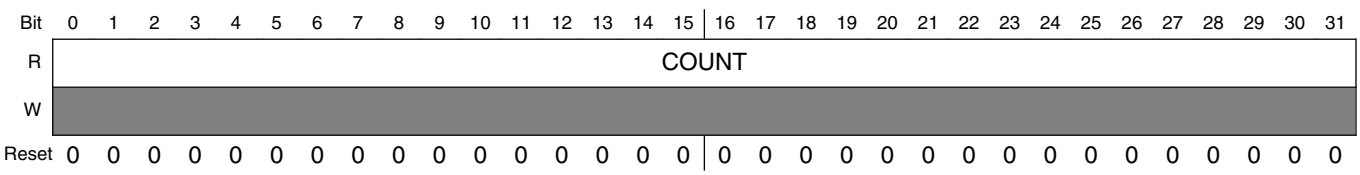


ENET_RMON_R_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of greater-than-2048-byte recieve packets

61.5.81 Rx Octets Statistic Register (ENET_RMON_R_OCTETS)

Address: 0h base + 2C4h offset = 2C4h



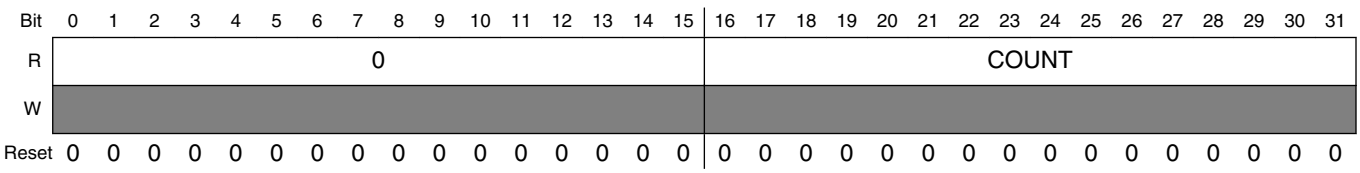
ENET_RMON_R_OCTETS field descriptions

Field	Description
0–31 COUNT	Number of receive octets

61.5.82 Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: 0h base + 2C8h offset = 2C8h

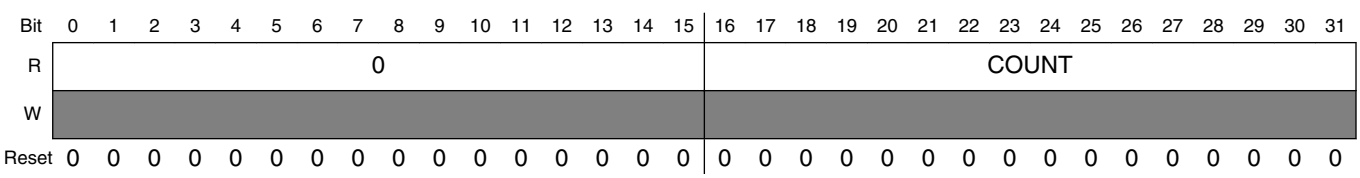


ENET_IEEE_R_DROP field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Frame count

61.5.83 Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)

Address: 0h base + 2CCh offset = 2CCh

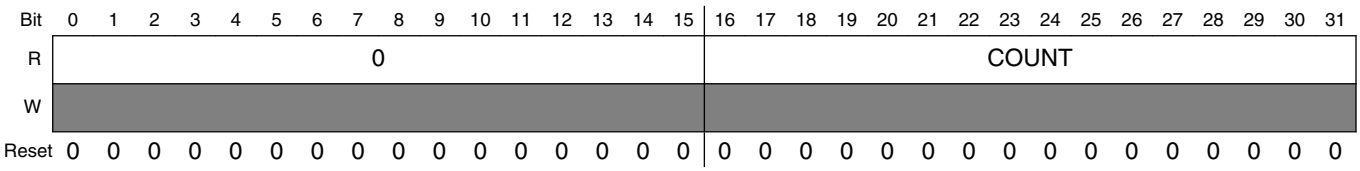


ENET_IEEE_R_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received OK

61.5.84 Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)

Address: 0h base + 2D0h offset = 2D0h

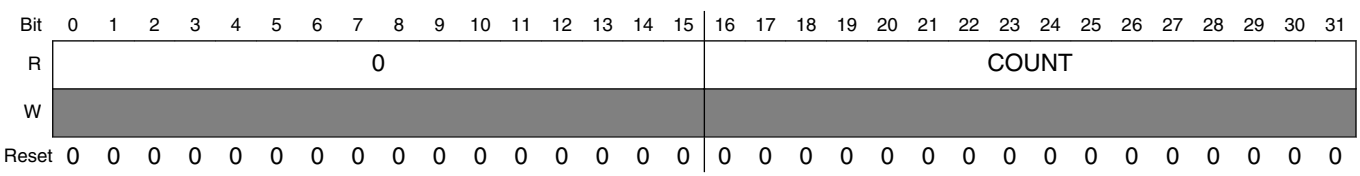


ENET_IEEE_R_CRC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with CRC error

61.5.85 Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)

Address: 0h base + 2D4h offset = 2D4h

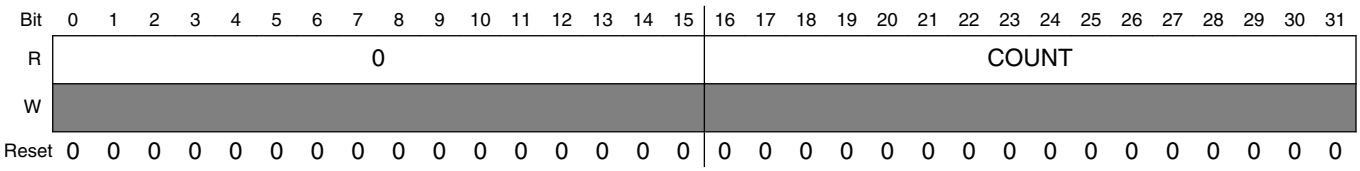


ENET_IEEE_R_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with alignment error

61.5.86 Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)

Address: 0h base + 2D8h offset = 2D8h

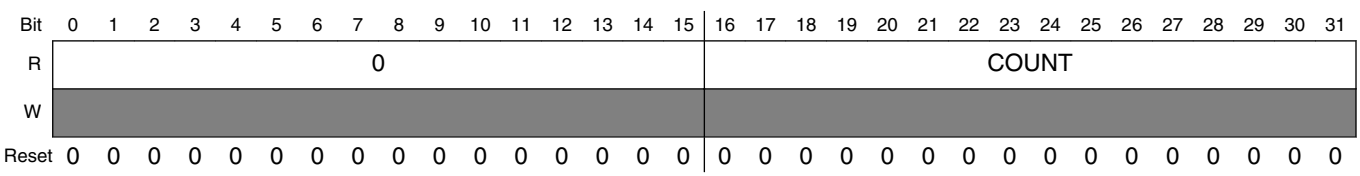


ENET_IEEE_R_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Receive FIFO overflow count

61.5.87 Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)

Address: 0h base + 2DCh offset = 2DCh



ENET_IEEE_R_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames received

61.5.88 Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

61.5.89 Adjustable Timer Control Register (ENET_ATCR)

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

NOTE

The CAPTURE and RESTART fields and bits 19 and 21 must be 0 in order to write to the other fields in this register.

Address: 0h base + 400h offset = 400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		SLAVE	Reserved	CAPTURE	Reserved	RESTART		PINPER			PEREN	OFFRST	OFFEN		EN
W								0		0	1				0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SLAVE	Enable Timer Slave Mode 0 The timer is active and all configuration fields in this register are relevant. 1 The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
19 Reserved	This field is reserved. Always write 0 to this field.
20 CAPTURE	Capture Timer Value When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. 0 No effect. 1 The current time is captured and can be read from the ATVR register.
21 Reserved	This field is reserved. Always write 0 to this field.
22 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes.
23 Reserved	This field is reserved.
24 PINPER	Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details. 0 Disable. 1 Enable.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved. **NOTE:** This field must be written always with one.
27 PEREN	Enable Periodical Event 0 Disable. 1 A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.
28 OFFRST	Reset Timer On Offset Event 0 The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1 If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.

Table continues on the next page...

ENET_ATCR field descriptions (continued)

Field	Description
29 OFFEN	Enable One-Shot Offset Event 0 Disable. 1 The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
30 Reserved	This field is reserved.
31 EN	Enable Timer 0 The timer stops at the current value. 1 The timer starts incrementing.

61.5.90 Timer Value Register (ENET_ATVR)

Address: 0h base + 404h offset = 404h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ATIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATVR field descriptions

Field	Description
0–31 ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

61.5.91 Timer Offset Register (ENET_ATOFF)

Address: 0h base + 408h offset = 408h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OFFSET																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATOFF field descriptions

Field	Description
0–31 OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

61.5.92 Timer Period Register (ENET_ATPER)

Address: 0h base + 40Ch offset = 40Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PERIOD																															
Reset	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

ENET_ATPER field descriptions

Field	Description
0–31 PERIOD	Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter. **NOTE:** The value of PERIOD has the following constraint: 2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

61.5.93 Timer Correction Register (ENET_ATCOR)

Address: 0h base + 410h offset = 410h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	COR														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

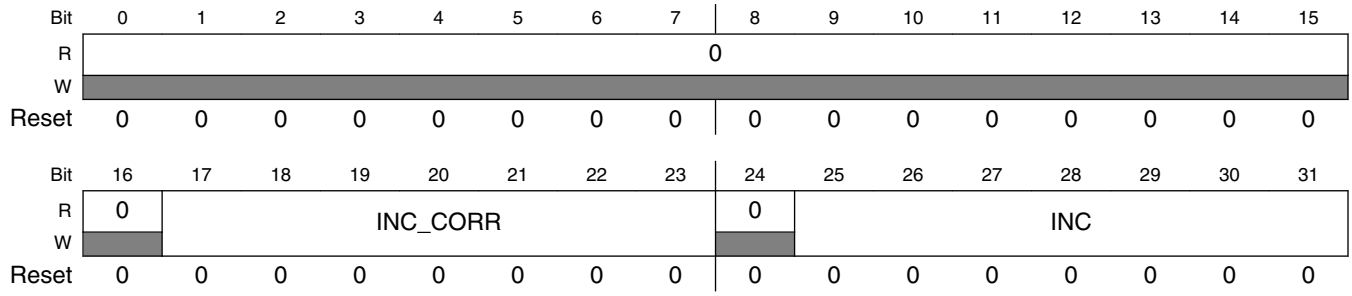
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	COR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCOR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–31 COR	Correction Counter Wrap-Around Value Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

61.5.94 Time-Stamping Clock Period Register (ENET_ATINC)

Address: 0h base + 414h offset = 414h

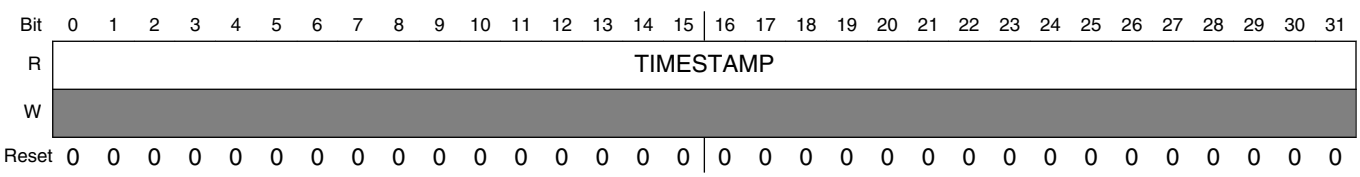


ENET_ATINC field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

61.5.95 Timestamp of Last Transmitted Frame (ENET_ATSTMP)

Address: 0h base + 418h offset = 418h



ENET_ATSTMP field descriptions

Field	Description
0–31 TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

61.5.96 Timer Global Status Register (ENET_TGSR)

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

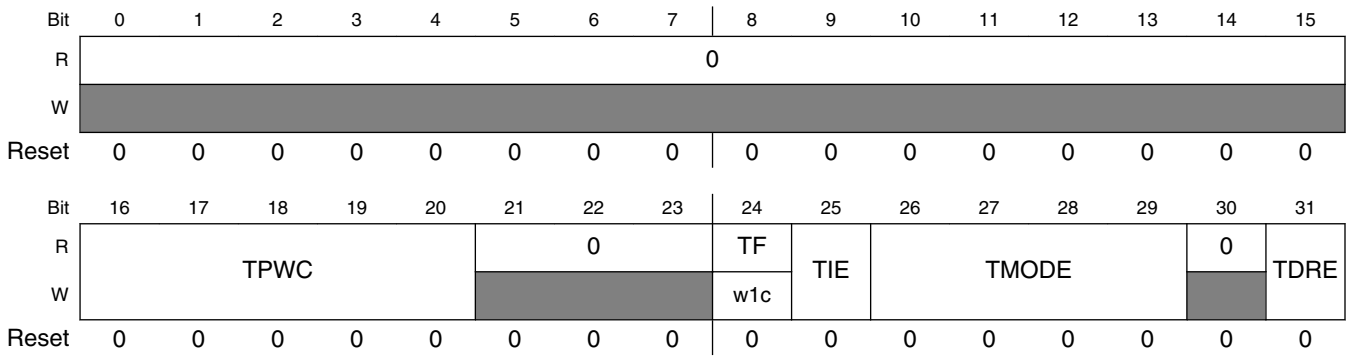
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												TF3	TF2	TF1	TF0
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TGSR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TF3	Copy Of Timer Flag For Channel 3 0 Timer Flag for Channel 3 is clear 1 Timer Flag for Channel 3 is set
29 TF2	Copy Of Timer Flag For Channel 2 0 Timer Flag for Channel 2 is clear 1 Timer Flag for Channel 2 is set
30 TF1	Copy Of Timer Flag For Channel 1 0 Timer Flag for Channel 1 is clear 1 Timer Flag for Channel 1 is set
31 TF0	Copy Of Timer Flag For Channel 0 0 Timer Flag for Channel 0 is clear 1 Timer Flag for Channel 0 is set

61.5.97 Timer Control Status Register (ENET_TCSR*n*)

Address: 0h base + 608h offset + (8d × i), where i=0d to 3d



ENET_TCSR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–20 TPWC	Timer PulseWidth Control Specifies the pulse width associated with TMODE values of 1110 or 11X1. Updating this field takes a few cycles to register because it is synchronized to the 1588 clock. When changing this field: 1. Always disable the channel and read the TMODE field to verify that the channel is disabled. 2. Set TPWC to the desired value. 3. Reenable the channel. 00000 Pulse width is one 1588-clock cycle. 00001 Pulse width is two 1588-clock cycles. 00010 Pulse width is three 1588-clock cycles. 00011 Pulse width is four 1588-clock cycles. ... ... 11111 Pulse width is 32 1588-clock cycles.
21–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0 Input Capture or Output Compare has not occurred. 1 Input Capture or Output Compare has occurred.
25 TIE	Timer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
26–29 TMODE	Timer Mode

Table continues on the next page...

ENET_TCSR*n* field descriptions (continued)

Field	Description
	Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000 Timer Channel is disabled. 0001 Timer Channel is configured for Input Capture on rising edge. 0010 Timer Channel is configured for Input Capture on falling edge. 0011 Timer Channel is configured for Input Capture on both edges. 0100 Timer Channel is configured for Output Compare - software only. 0101 Timer Channel is configured for Output Compare - toggle output on compare. 0110 Timer Channel is configured for Output Compare - clear output on compare. 0111 Timer Channel is configured for Output Compare - set output on compare. 1000 Reserved 1010 Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10X1 Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110X Reserved 1110 Timer Channel is configured for Output Compare - pulse output low on compare for 1 to 32 1588-clock cycles as specified by TPWC. 1111 Timer Channel is configured for Output Compare - pulse output high on compare for 1 to 32 1588-clock cycles as specified by TPWC.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDRE	Timer DMA Request Enable 0 DMA request is disabled 1 DMA request is enabled

61.5.98 Timer Compare Capture Register (ENET_TCCR*n*)

Address: 0h base + 60Ch offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TCC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCCR*n* field descriptions

Field	Description
0–31 TCC	Timer Capture Compare This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag.

ENET_TCCR*n* field descriptions (continued)

Field	Description
	The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag.

61.6 Functional description

This section provides a complete functional description of the MAC-NET core.

61.6.1 Ethernet MAC frame formats

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field
- Frame check sequence (CRC value)
- Extension field is defined only for Gigabit Ethernet half-duplex implementations and is not supported by the MAC core

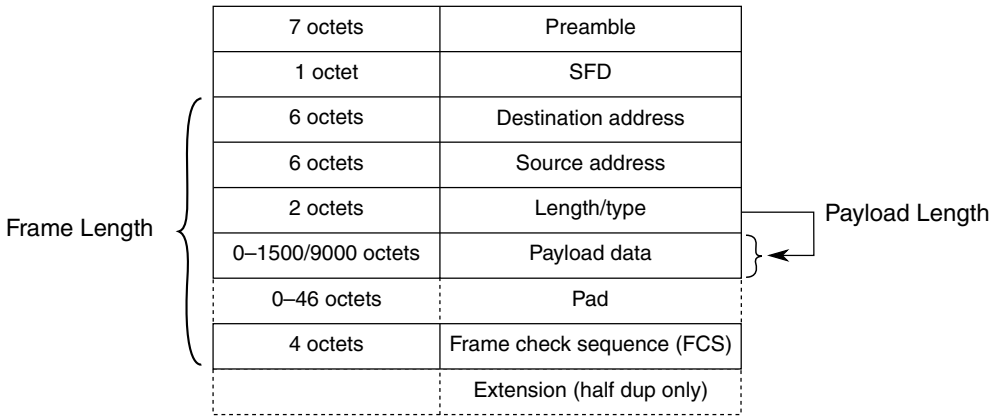


Figure 61-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

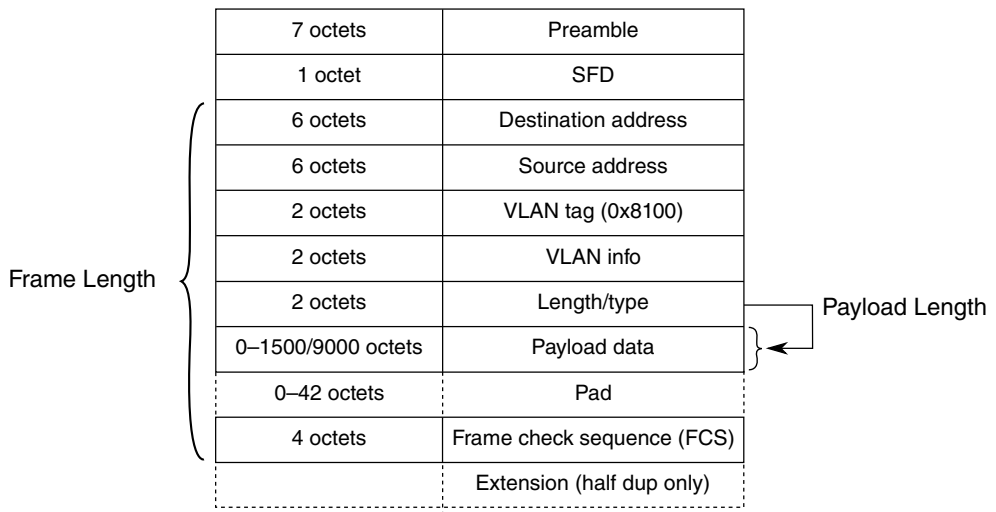


Figure 61-3. VLAN-tagged MAC frame format overview

Table 61-6. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first. • If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.

Table continues on the next page...

Table 61-6. MAC frame definition (continued)

Term	Description
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

61.6.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 61-7. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

61.6.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

61.6.2 IP and higher layers frame format

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

61.6.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 61-8. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

61.6.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 61-9. IPv4 header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Version	IHL	TOS		Length			
Fragment ID				Flags	Fragment offset		
TTL		Protocol		Header checksum			
Source address							
Destination address							
Options							

Table 61-10. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

61.6.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the

IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

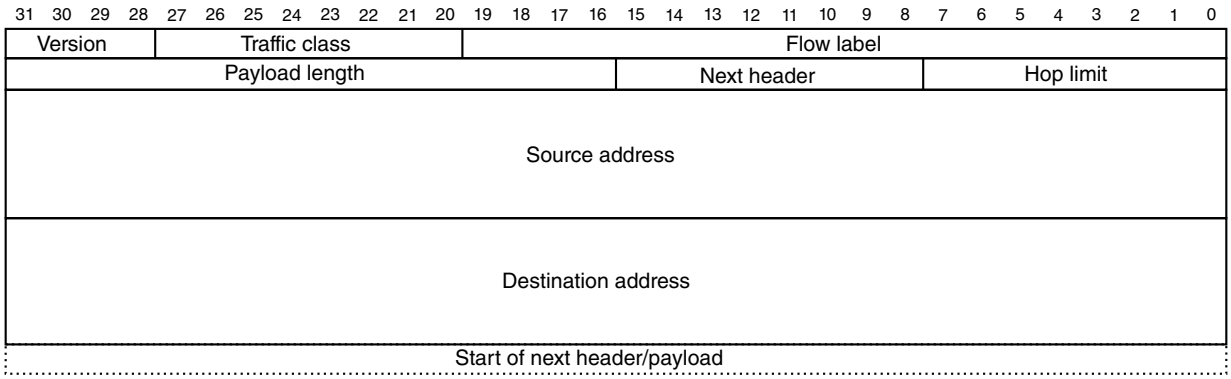


Figure 61-4. IPv6 header format

Table 61-11. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

61.6.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 61-12. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 61-13. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

61.6.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 61-14. UDP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Length				Checksum			

Table 61-15. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

61.6.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 61-16. TCP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			

Table continues on the next page...

Table 61-16. TCP header format (continued)

Sequence number			
Acknowledgement number			
Offset	Reserved	Flags	Window
Checksum			Urgent pointer
Options			

Table 61-17. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

61.6.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

61.6.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

61.6.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

Table 61-18 shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 61-18. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 61-19. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

61.6.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 61-20. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

61.6.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

61.6.3.2.1 PTPv1 header

Table 61-21. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 61-22 :

Table 61-22. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

61.6.3.2.2 PTPv2 header

Table 61-23. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 61-24. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 61-25. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

61.6.4 MAC receive

- Check frame framing
- Remove frame preamble and frame SFD field
- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists
- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

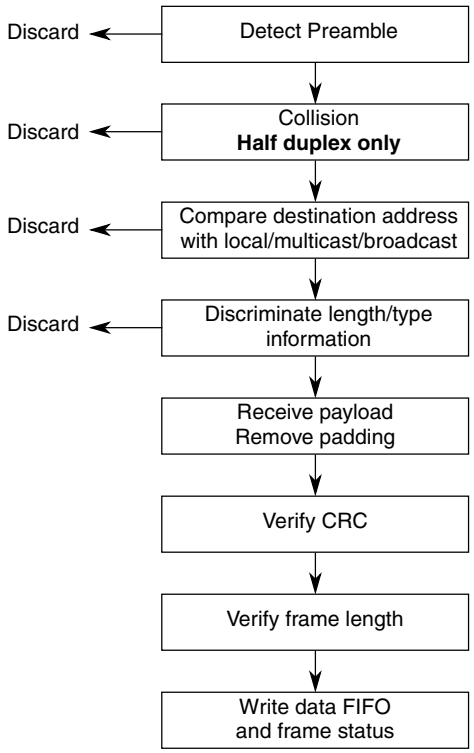


Figure 61-5. MAC receive flow

61.6.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

61.6.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

61.6.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

61.6.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

61.6.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).
- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

61.6.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 61-26. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

61.6.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set
- A unicast is received that does not match either:

- Node address (PALR[PADDR1] and PAUR[PADDR2])
- Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])

- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

61.6.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word and interrupt bit PLR.

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

61.6.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

61.6.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[NO]) indicates that the current frame is VLAN tagged.

61.6.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Good pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

61.6.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is set.

When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

61.6.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the

padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

61.6.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

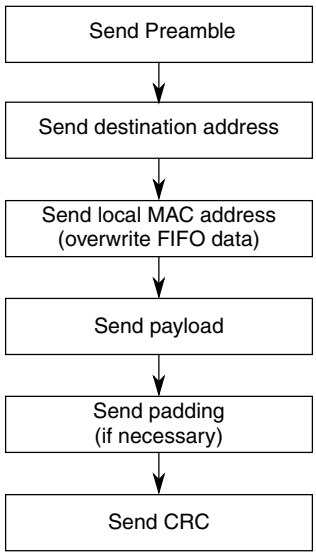


Figure 61-6. Frame transmit overview

61.6.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

61.6.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

61.6.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

61.6.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS/GMII_CRS de-assertion.

61.6.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

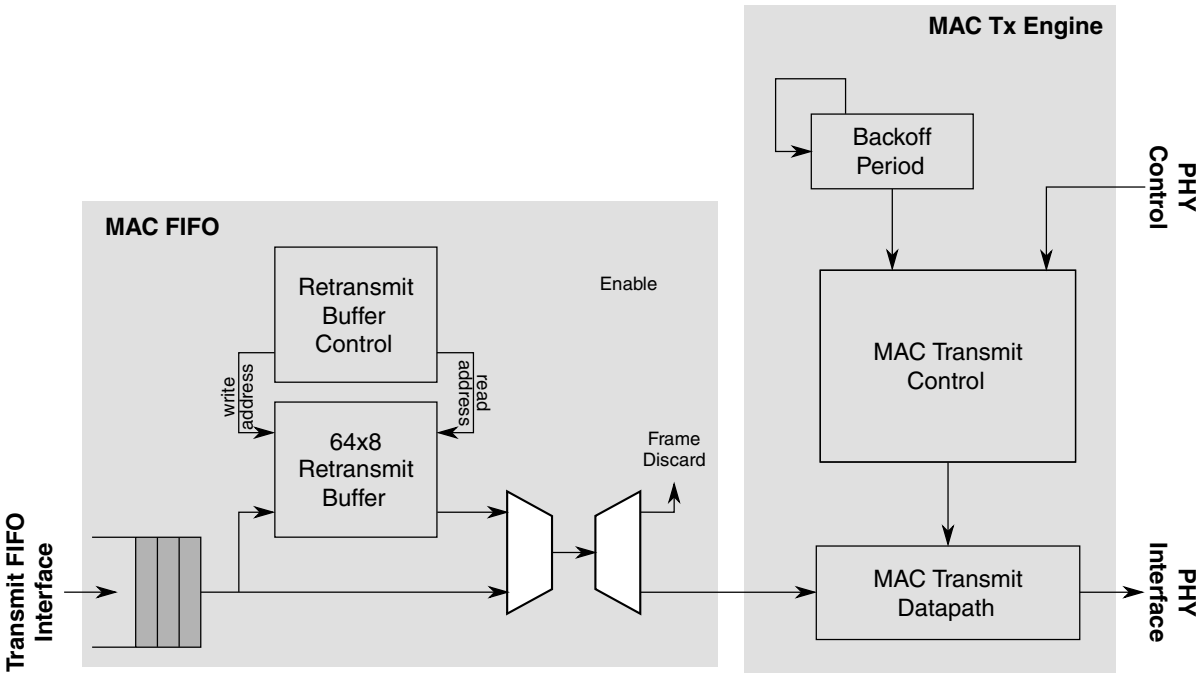


Figure 61-7. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the n^{th} re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- $k = \min(n, N)$; where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

Ethernet PHYs that support the SQE Test, or "heartbeat,"
feature must disable this feature. When this feature is enabled,

the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

61.6.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

61.6.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.
- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

61.6.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

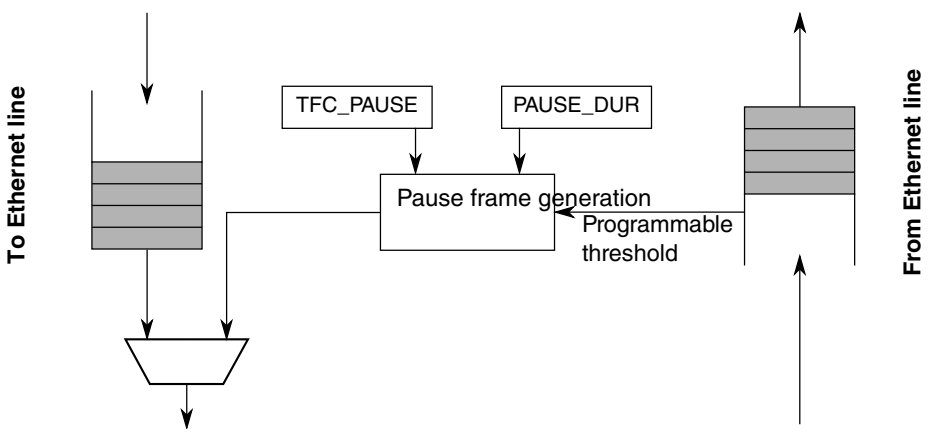


Figure 61-8. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

61.6.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

61.6.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, when the processor is in Stop mode, Sleep mode is entered, without affecting the ENET*n*_ECR register bits.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

61.6.7.2 Magic packet detection

The core is designed to detect magic packets with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

61.6.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

61.6.8 IP accelerator functions

61.6.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 61-27. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero			Protocol		TCP/UDP length		

Table 61-28. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

61.6.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

61.6.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 61-29.

Table 61-29. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

61.6.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

61.6.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

61.6.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL
  - Frame received with a CRC-32 error
  - Frame truncated due to receive FIFO overflow
  - Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)
- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

61.6.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

61.6.8.6 IPv6 support

The following sections describe the IPv6 support.

61.6.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing.

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

61.6.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

61.6.9 Resets and stop controls

61.6.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

61.6.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting GMII/TXER to the PHY.

- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

61.6.9.3 Hardware freeze

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII/GMII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

61.6.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

61.6.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

61.6.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII/GMII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set

- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
  - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
  - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

61.6.9.4.3 Graceful stop interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

61.6.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

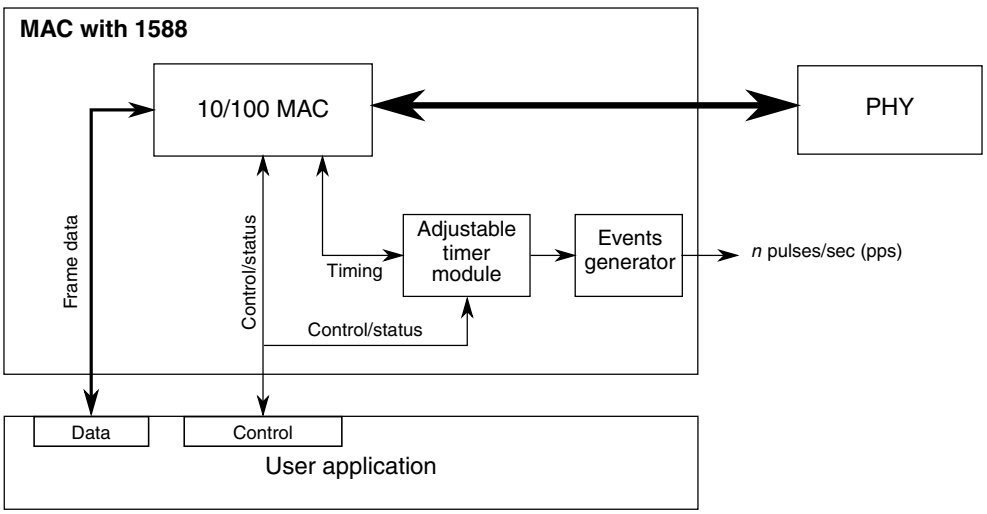


Figure 61-9. IEEE 1588 functions overview

61.6.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1×10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

61.6.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

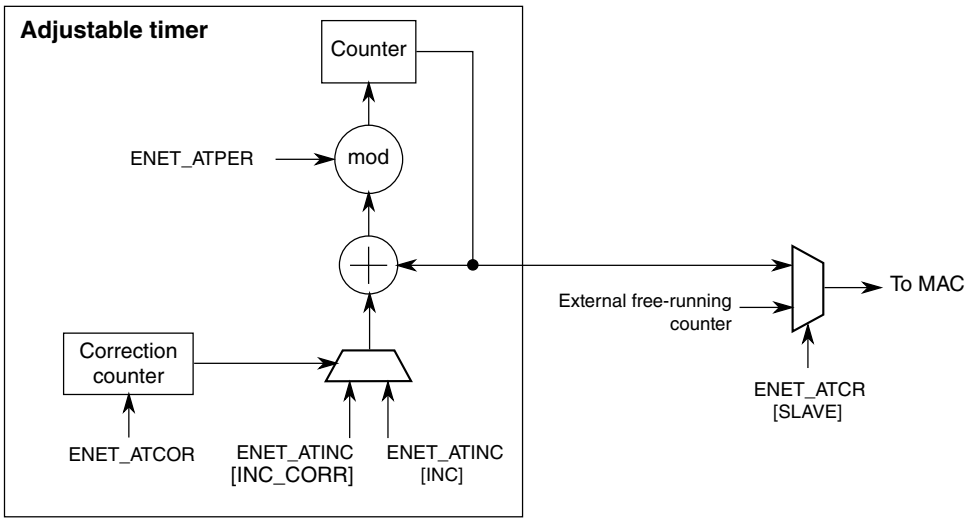


Figure 61-10. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc $-(1 \times 10^9)$), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules

within the system must operate from a single timer. When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

61.6.10.2 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

61.6.10.3 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

61.6.10.4 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to (1×10^9)–1. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

61.6.10.5 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

61.6.10.5.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

61.6.10.5.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

61.6.10.5.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

61.6.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 512 words.

61.6.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation.

Table 61-30. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM [RX_ALMOST_EMPTY]	When the FIFO level reaches the ENET*n*_RAEM value, and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

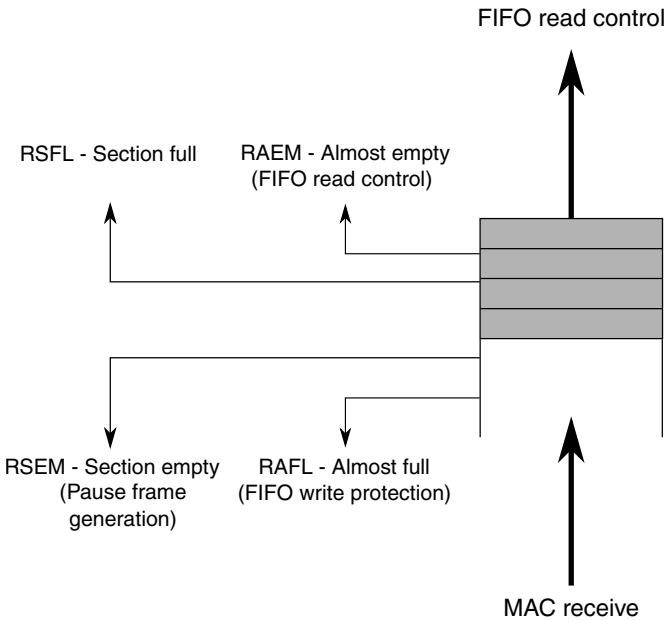


Figure 61-11. Receive FIFO overview

61.6.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation.

Table 61-31. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an GMII/MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation).

Table 61-31. Transmit FIFO thresholds definition

Register	Description
	If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

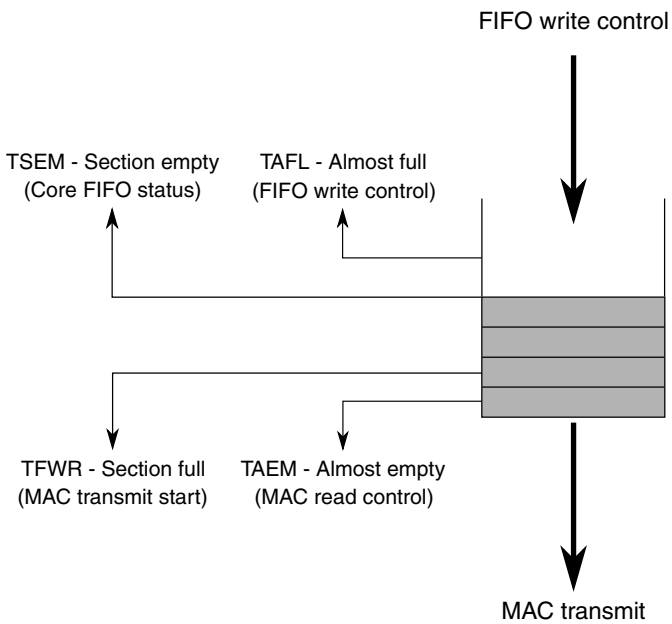


Figure 61-12. Transmit FIFO overview

61.6.12 Loopback options

The core implements external and internal loopback options, which are controlled by the ENET*n*_RCR register fields found here.

Table 61-32. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

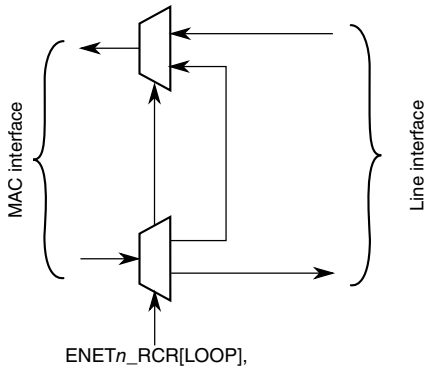


Figure 61-13. Loopback options

61.6.13 Legacy buffer descriptors

To support the Ethernet controller on previous chips, legacy FEC buffer descriptors are available. To enable legacy support, write 0 to ENET*n*_ECR[1588EN].

NOTE

- The legacy buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

61.6.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 61-36 contains the descriptions for each field.

Table 61-33. Legacy FEC receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer — high halfword															
Offset + 6	Rx data buffer pointer — low halfword															

61.6.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 61-38 contains the descriptions for each field.

Table 61-34. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer — high halfword															
Offset + 6	Tx Data Buffer Pointer — low halfword															

1. This field is not supported by the uDMA.

61.6.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

The enhanced buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

61.6.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 61-36 contains the descriptions for each field.

Table 61-35. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer – high halfword															
Offset + 6	Rx data buffer pointer – low halfword															
Offset + 8	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—

Table continues on the next page...

Table 61-35. Enhanced uDMA receive buffer descriptor (RxBD) (continued)

Offset + A	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + C	Header length					—	—	—	Protocol type							
Offset + E	Payload checksum															
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 61-36. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 0	3 RO2	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	4 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5–6	Reserved, must be cleared.
Offset + 0	7 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM fields are set to 1. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode. The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 0	8 BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).

Table continues on the next page...

Table 61-36. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 0	9 MC	Set if the DA is multicast and not BC.
Offset + 0	10 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 0	11 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 0	12	Reserved, must be cleared.
Offset + 0	13 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 0	14 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 0	15 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 2	0–15 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
0ffset + 4	0–15 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 6	0–15 Data buffer pointer low	Receive data buffer pointer, low halfword
Offset + 8	0 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + 8	1–4	Reserved, must be cleared.
Offset + 8	5 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + 8	6 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + 8	7 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + 8	8 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + 8	9–15	Reserved, must be cleared.

Table continues on the next page...

Table 61-36. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + A	0–2 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + A	3–9	Reserved, must be cleared.
Offset + A	10 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + A	11 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + A	12	Reserved, must be cleared.
Offset + A	13 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.
Offset + A	14 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + A	15 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.
Offset + C	0–4 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + C	5–7	Reserved, must be cleared.
Offset + C	8–15 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + E	0–15 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16		
Offset + 18	0–15	Reserved, must be cleared.

Table 61-36. Receive buffer descriptor field definitions

Word	Field	Description
– Offset + 1E		

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 64. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

61.6.14.2 Enhanced transmit buffer descriptor

Table 61-37. Enhanced transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 2	Data length															
Offset + 4	Tx Data Buffer Pointer – high halfword															
Offset + 6	Tx Data Buffer Pointer – low halfword															
Offset + 8	—	INT	TS	PINS	IINS	—	—	—	—	—	—	—	—	—	—	—
Offset + A	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 61-38. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 R	Ready. Written by the MAC and user. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 0	1	Transmit software ownership. This field is reserved for software use. This read/write field is not modified by hardware and its value does not affect hardware.

Table continues on the next page...

Table 61-38. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
	TO1	
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	3 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 0	4 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	5 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 0	6 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 0	7–15	Reserved, must be cleared.
Offset + 2	0–15 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 4	0–15 Data buffer pointer high	Tx data buffer pointer, high halfword. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller. **NOTE:** For optimal performance, make the transmit buffer pointer evenly divisible by 64.
Offset + 6	0–15 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 8	0	Reserved, must be cleared.
Offset + 8	1 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + 8	2 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + 8	3 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	4 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by

Table continues on the next page...

Table 61-38. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
		the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	5–15	Reserved, must be cleared.
Offset + A	0 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + A	1	Reserved, must be cleared.
Offset + A	2 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.
Offset + A	3 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.
Offset + A	4 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + A	5 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + A	6 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + A	7 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + A	8–15	Reserved, must be cleared.
Offset + C	0–15	Reserved, must be cleared.
Offset + E	0–15	Reserved, must be cleared.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16	1588 timestamp	
Offset + 18–Offset + 1E	0–15	Reserved, must be cleared.

61.6.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

61.6.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 61-39. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address
- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 61-40. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

Table 61-41. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

The standard defines that the LSB of the MAC address is sent/received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

61.6.15.2 Data structure examples

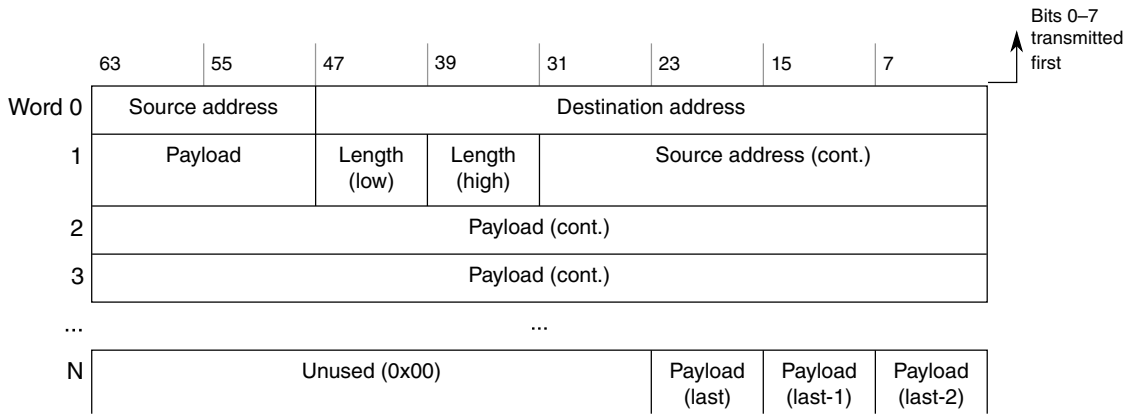


Figure 61-14. Normal Ethernet frame 64-bit mapping example

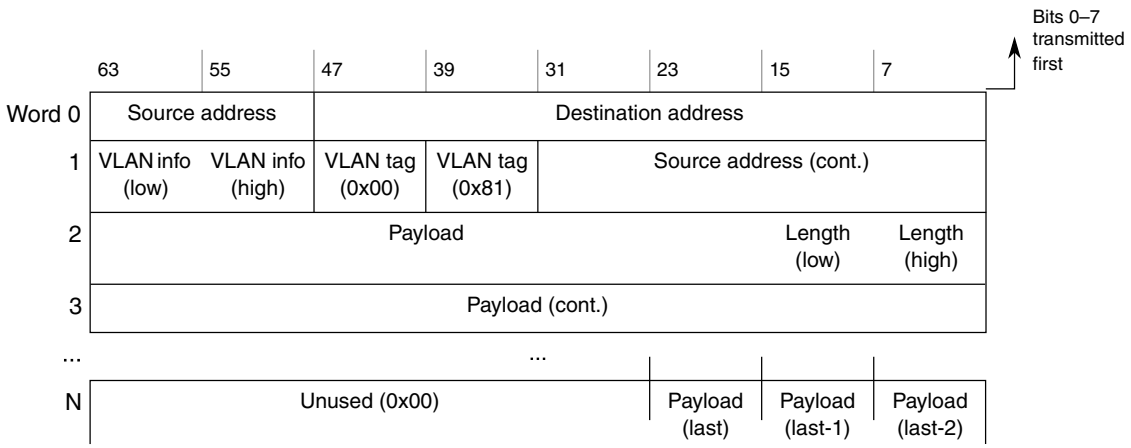


Figure 61-15. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

61.6.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

61.6.16 FIFO protection

61.6.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 61-16) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

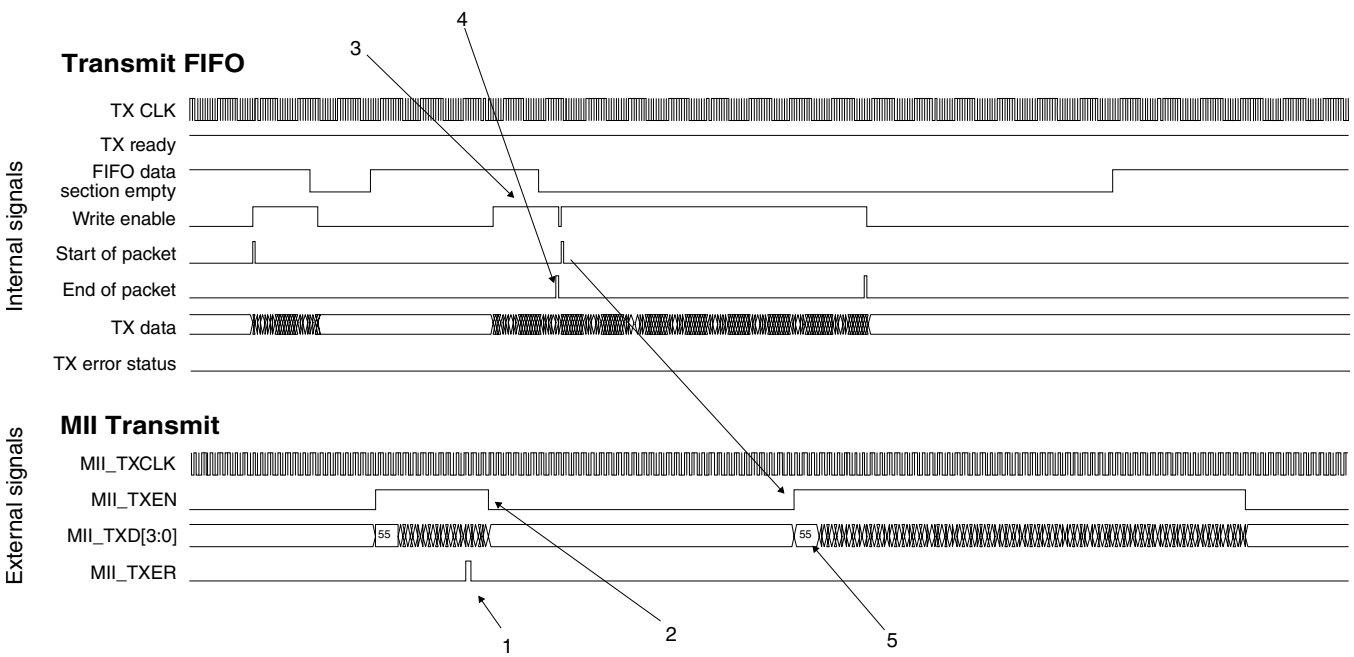


Figure 61-16. Transmit FIFO underflow protection

61.6.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

61.6.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

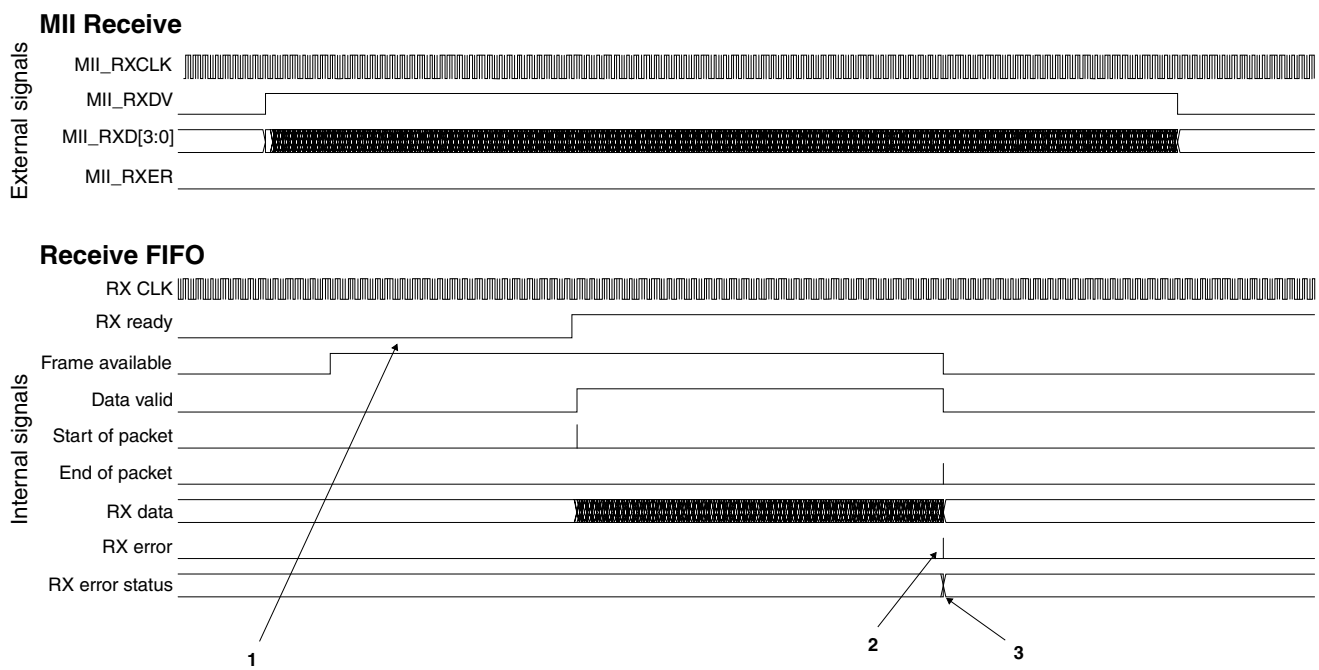


Figure 61-17. Receive FIFO overflow protection

61.6.17 Reference clock

The input clocks to the ENET module must meet the specifications in the following table.

Ethernet speed mode	Ethernet bus clock	Minimum ENET system clock
10 Mbit/s	2.5 MHz	5 MHz
100 Mbit/s	25.0 MHz	50 MHz
1000 Mbit/s[1]	125.0 MHz	133 MHz

1. Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.

61.6.18 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

61.6.18.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 61-42. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 61-43. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.

Table continues on the next page...

Table 61-43. MDIO frame field descriptions (continued)

Field	Description
(5 bits)	
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data (16 bits)	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

61.6.18.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 61-44. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 61-45. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 61-46 for more information. 00 Address write 01 Write operation

Table continues on the next page...

Table 61-45. MDIO clause 45 frame field descriptions (continued)

Field	Description
	10 Read inc. operation 11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 61-46. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

61.6.18.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

61.6.18.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

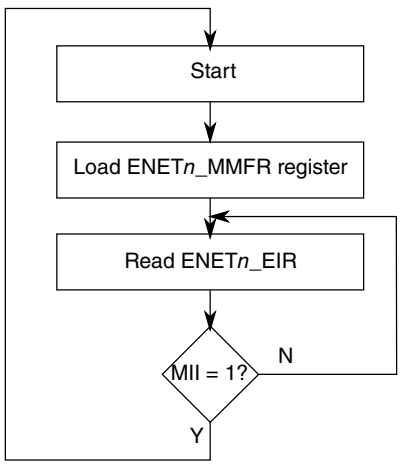


Figure 61-18. MDIO access overview

61.6.19 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets
- RGMII 10/100/1000 by way of interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	ECR[SPEED]	RCR[RMII_10T]	RCR[RMII_MODE]	RCR[RGMII_EN]
MII - 10 Mbit/s	0	—	0	0
MII - 100 Mbit/s	0	—	0	0
RMII - 10 Mbit/s	0	1	1	0
RMII - 100 Mbit/s	0	0	1	0
RGMII - 10 Mbit/s	0	1	0	1
RGMII - 100 Mbit/s	0	0	0	1
RGMII - 1000 Mbit/s[1]	1	—	0	1

1. Actual throughput is greater than 100 Mbit/s but significantly less than 1000 Mbit/s.

61.6.19.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

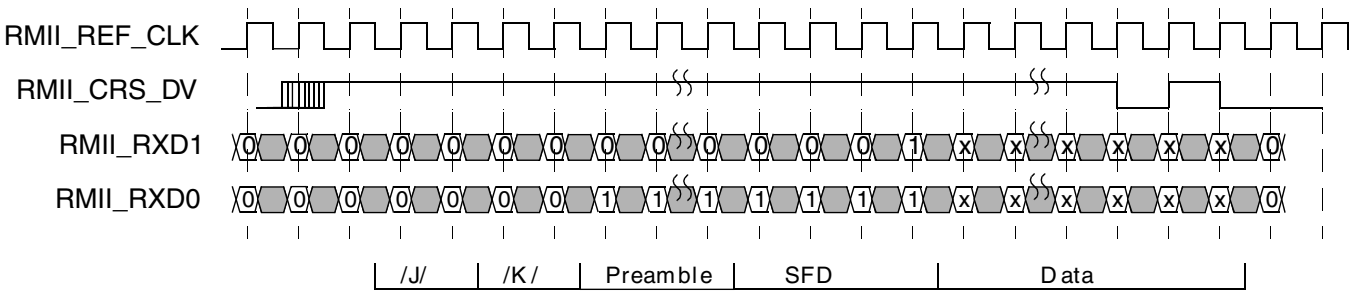


Figure 61-19. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

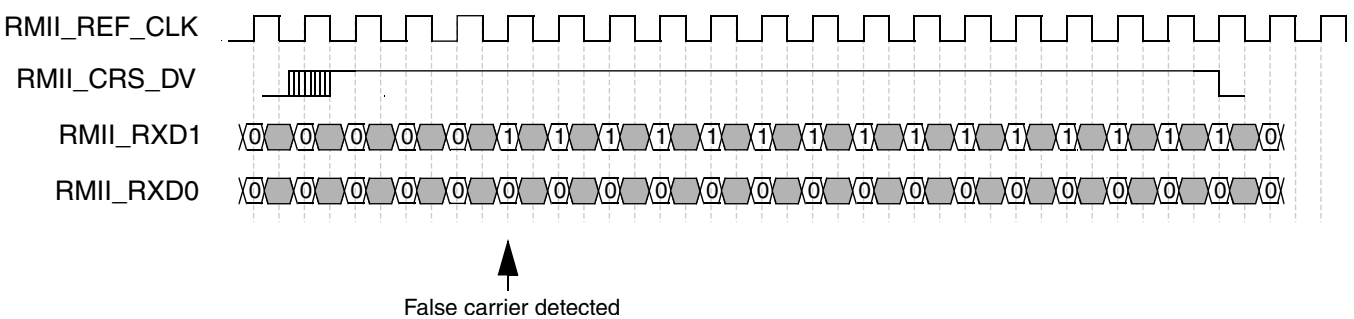


Figure 61-20. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

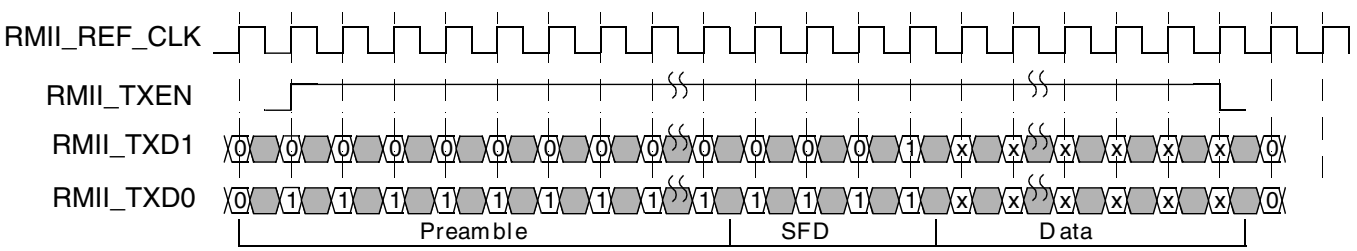


Figure 61-21. RMII transmit operation

61.6.19.2 RGMII interface

In RGMII modes, the data and control information is multiplexed by taking advantage of both edges of the reference clocks.

The data signals contain the lower four data bits on the rising edge and the upper four bits on the falling edge. The control signals are multiplexed into a single clock cycle using the same technique.

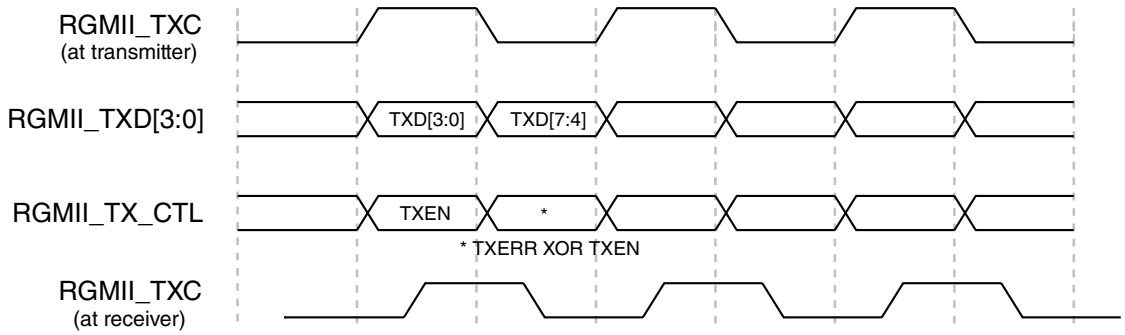


Figure 61-22. RGMII transmit operation

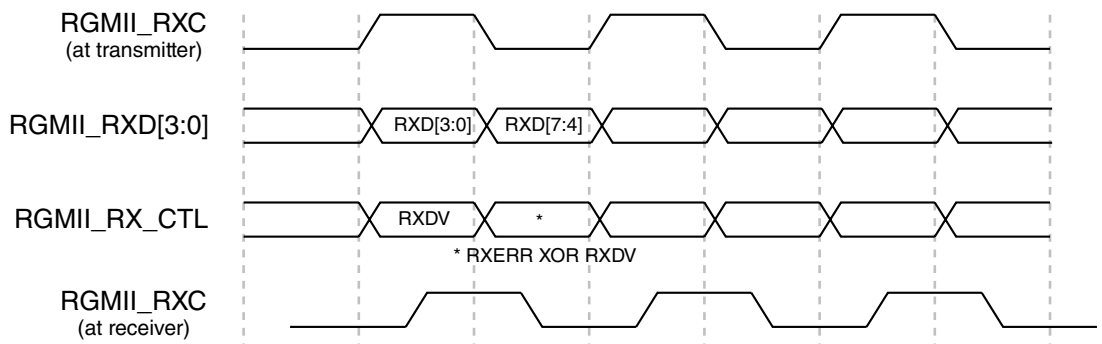


Figure 61-23. RGMII receive operation

61.6.19.3 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

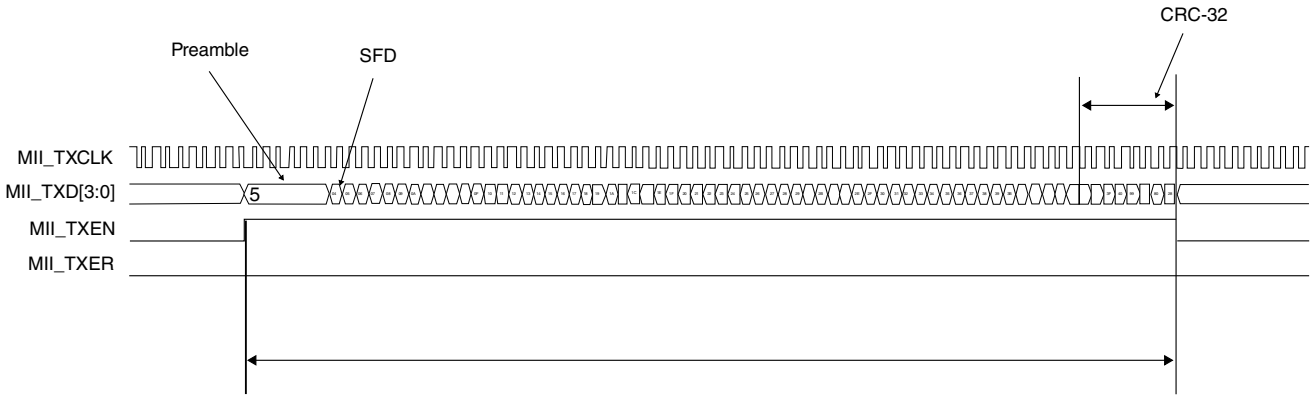


Figure 61-24. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

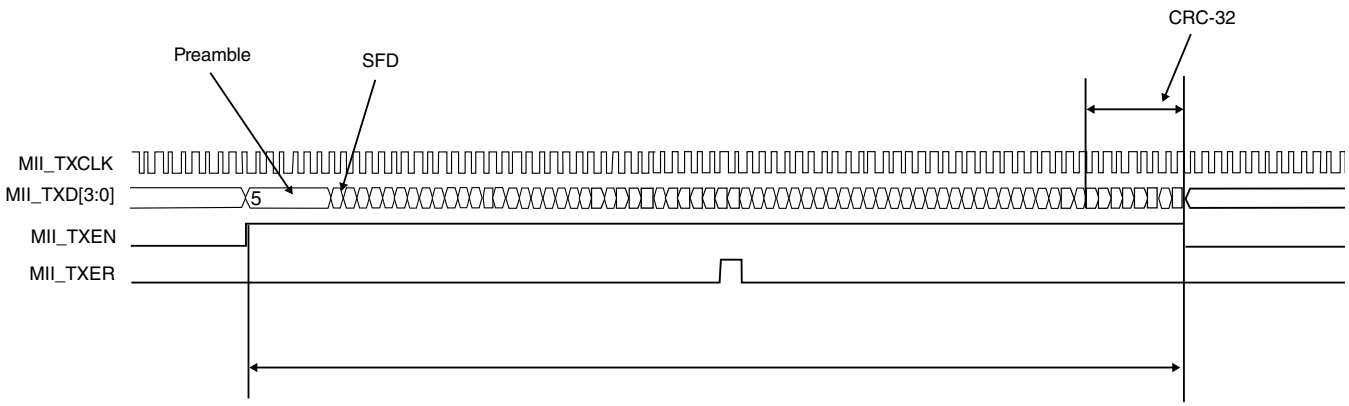


Figure 61-25. MII transmit operation — errored frame

61.6.19.3.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

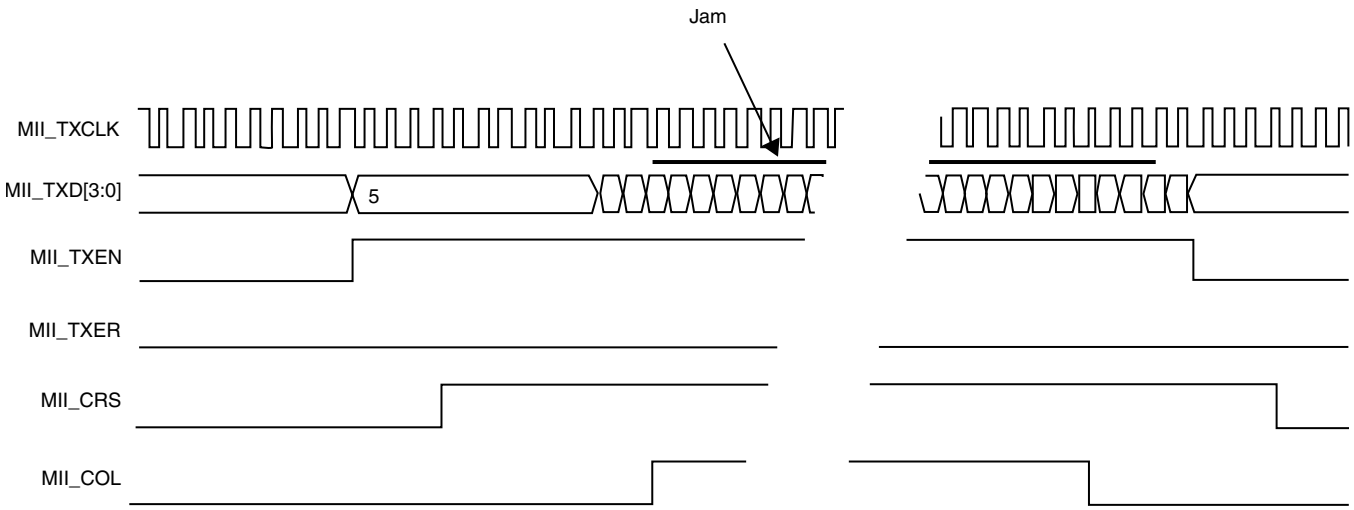


Figure 61-26. MII transmit operation — transmission with collision

61.6.19.4 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

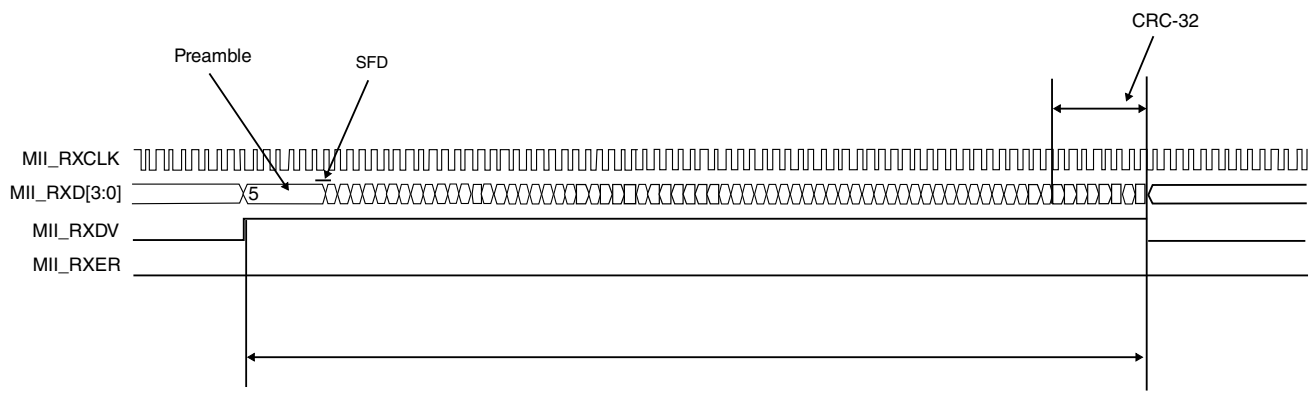


Figure 61-27. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

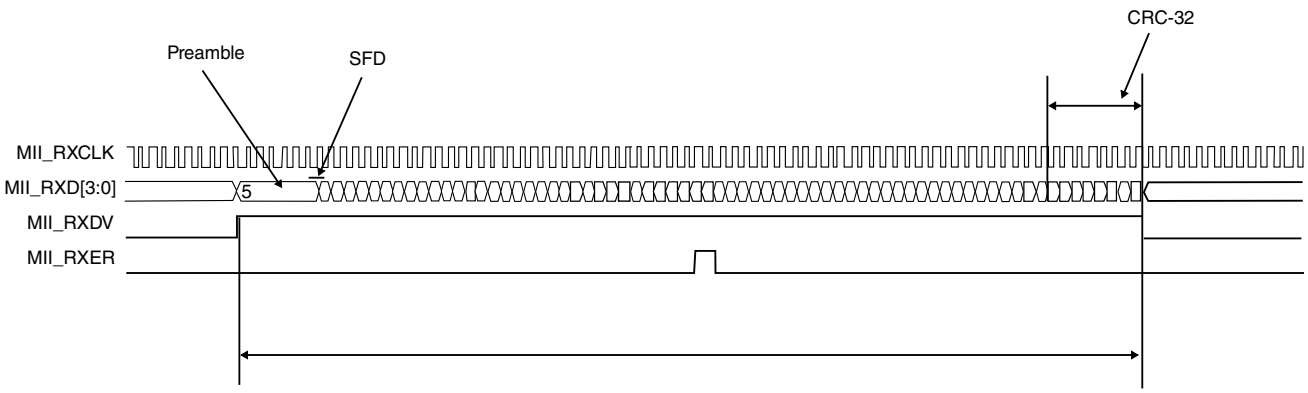


Figure 61-28. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

61.6.20 Interrupt coalescence

The purpose of the interrupt coalescing is to reduce the number of interrupts generated by the MAC so as to reduce the CPU loading.

To facilitate this interrupt coalescing, these registers are available with the same control and configuration fields.

- Transmit Interrupt Coalescing Register (ENET_TXIC)
- Receive Interrupt Coalescing Register (ENET_RXIC)

When coalescing is enabled by asserting the corresponding ICEN field and such interrupt is also enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt when the threshold number of frames is reached (defined by ICFT) or when the threshold timer expires (defined by ICTT).

When coalescing is disabled by de-asserting ICEN, but interrupt is enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt as they are received without using coalescing. Interrupt coalescing is done for each transmit and receive queue/class independently.

61.6.20.1 Interrupt coalescence setup

Interrupt coalescence supports both legacy and enhanced BDs. The following guidelines are recommended when setting up interrupt coalescence.

- When the MAC is configured for enhanced (IEEE 1588) mode, that is, enhanced BDs:
  - Set the INT bit in the enhanced received buffer descriptor to one.
  - Set the INT bit in the enhanced transmit buffer descriptor(s) to one.
- Clear the TXB and RXB fields in the EIMR register.

61.6.20.2 Updating the frame count threshold on-the-fly

To update the ICFT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new threshold value to the ICFT field.
3. Set ICEN to one.

NOTE

The ICFT field can be updated on-the-fly without disabling the ICEN field. The hardware interrupt will continue and there is a possibility that an interrupt will occur depending on the state of the hardware counter and the previous ICFT value.

61.6.20.3 Updating the timer threshold on-the-fly

To update the ICTT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new timer value to the ICTT field.
3. Set ICEN to one.

Chapter 62
Reset Generation Module (MC_RGM)

62.1 Introduction

62.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (***PHASE0***, ***PHASE1***, ***PHASE2***, ***PHASE3***, and ***IDLE***) of the reset sequence and controls the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

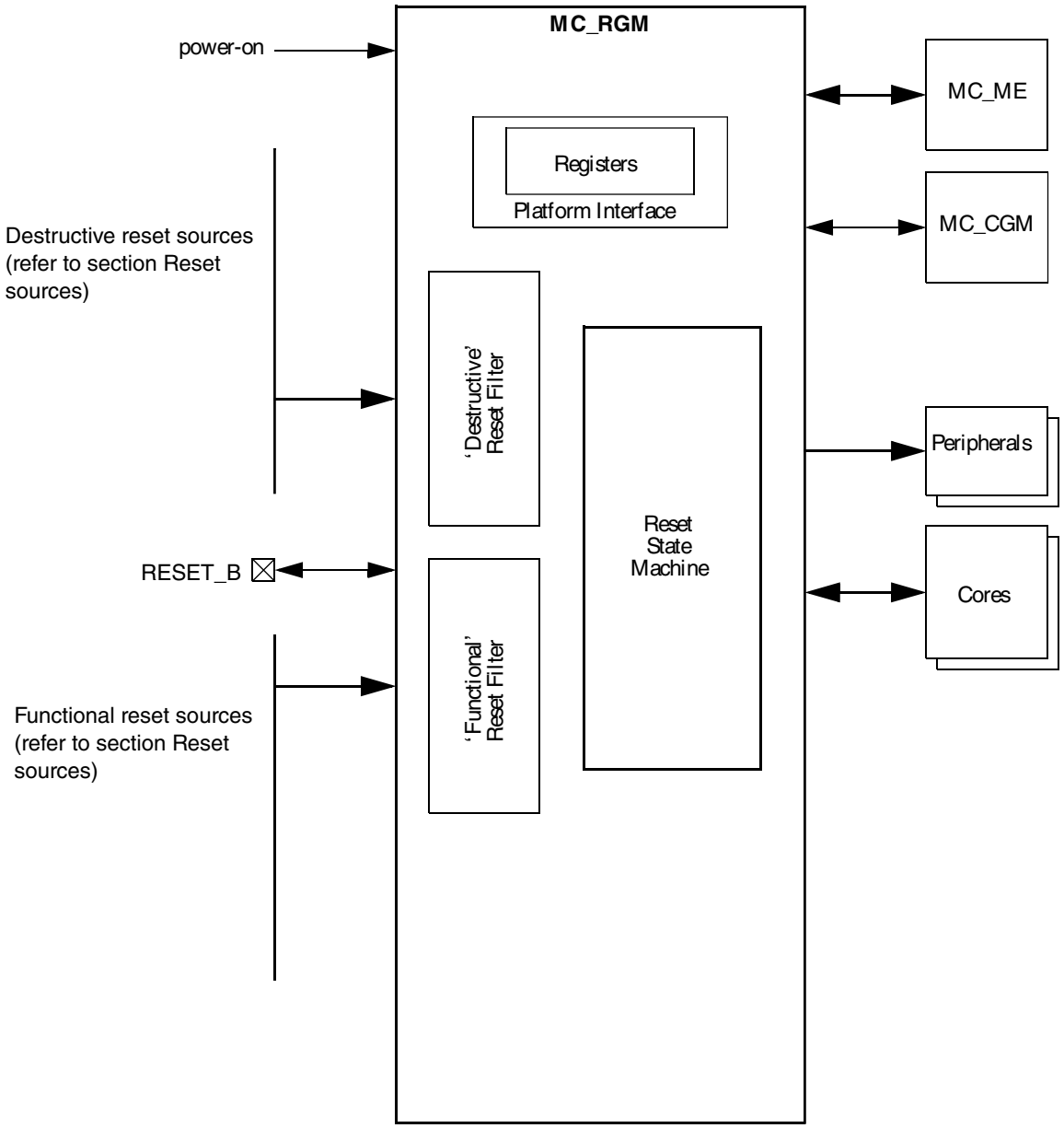


Figure 62-1. MC_RGM block diagram

62.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to ***SAFE mode*** or interrupt request events
- short reset sequence configuration
- bidirectional reset behavior configuration

- configurable escalation of recurring 'functional' resets to 'destructive' reset
- configurable escalation of recurring ‘destructive’ resets to keep chip in reset state until next power-on reset

62.2 External signal description

The MC_RGM interfaces to the reset pin RESET_B.

62.3 Reset sources

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

A ‘destructive’ reset source is associated with an event related to a critical—usually hardware—error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the chip starting from ***PHASE0***. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown except in the software ‘destructive’ reset case, which does preserve the system memory content if the chip is not configured to execute a self-test on power-on, ‘destructive’, and external resets. ‘Destructive’ resets are:

- power-on reset
- software ‘destructive’ reset
- FCCU failure to react reset
- ‘functional’ reset escalation
- temperature sensor 'destructive' reset
- voltage out of range ‘destructive’ resets
- STCU critical fault reset
- AFE LVDs
- MIPICSI2 LVDs
- SSCM secure reset request

NOTE

Each destructive reset also asserts external reset pin.

A ‘functional’ reset source is associated with an event related to a less critical—usually non-hardware—error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the chip starting from ***PHASE1***. In this case, most digital modules are reset normally, and the memory content must be considered to be unknown, while the state of analog modules or specific digital modules (e.g., debug modules, flash

modules) content is preserved, except on an external reset in the case where the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets.
‘Functional’ resets are

- External reset
- STCU self test completed
- Software ‘functional’ reset
- FCCU hard reaction
- FCCU soft reaction
- JTAG ‘functional’ reset
- Temperature sensor 'functional' reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., ***PHASEn*** states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the ***IDLE*** phase. During this entire process, the MC_ME state machine is held in ***RESET*** mode. Only at the end of the reset sequence, when the ***IDLE*** phase is reached, does the MC_ME enter the ***DRUN*** mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt issued to the core (see ‘Destructive’ Event Reset Disable Register (RGM_DERD) and ‘Destructive’ Event Alternate Request Register (RGM_DEAR) for ‘destructive’ resets and ‘Functional’ Event Reset Disable Register (RGM_FERD) and ‘Functional’ Event Alternate Request Register (RGM_FEAR) for ‘functional’ resets).

62.4 Memory map and register definition

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

NOTE

Reserved fields should always be written with their reset values only.

The bytes are ordered according to big endian. For example, the **RGM_DES[7:0]** register bits may be accessed as a word at address offset 0x000, as a half-word at address offset offset 0x002, or as a byte at address offset 0x003. Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behaviour cannot be changed.

NOTE

The following registers are reset to their default values during a POR event only:

- RGM_DES
- RGM_DERD
- RGM_DEAR
- RGM_FES
- RGM_DRET

RGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	'Destructive' Event Status Register (RGM_DES)	32	R/W	0000_0001h	62.4.1/3287
10	'Destructive' Event Reset Disable Register (RGM_DERD)	32	R/W	0000_0000h	62.4.2/3291
20	'Destructive' Event Alternate Request Register (RGM_DEAR)	32	R/W	0000_0000h	62.4.3/3294
300	'Functional' Event Status Register (RGM_FES)	32	R/W	0000_0000h	62.4.4/3295
310	'Functional' Event Reset Disable Register (RGM_FERD)	32	R/W	0000_0000h	62.4.5/3297
320	'Functional' Event Alternate Request Register (RGM_FEAR)	32	R/W	0000_0000h	62.4.6/3299
330	'Functional' Bidirectional Reset Enable Register (RGM_FBRE)	32	R/W	0180_0468h	62.4.7/3301
340	'Functional' Event Short Sequence Register (RGM_FESS)	32	R/W	0000_0040h	62.4.8/3303
604	'Functional' Event Short Sequence Register (RGM_FRET)	8	R/W	0Fh	62.4.9/3305
608	Destructive Reset Escalation Threshold Register (RGM_DRET)	8	R/W	00h	62.4.10/ 3306

62.4.1 'Destructive' Event Status Register (RGM_DES)

This register can be accessed as read/write in either supervisor mode or test mode and accessed as read only in user mode. Register bits are cleared on write '1'. This register is reset only on power-on.

NOTE

If LVD/HVD event is disabled and RGM is configured to issue an interrupt instead of reset, operation below LVD or above HVD levels is not guaranteed. User should guarantee that if LVD/HVD is disabled, then the voltage input is always in range.

NOTE

When RGM_DES[F_POR] is set the user needs to ignore all other bits in this register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			F_LVD_MIPI	F_AFE_LVD9	F_AFE_LVD6	F_AFE_LVD8	F_AFE_LVD7	F_TSR_DEST	F_AFE_LVD5	F_AFE_LVD4	F_AFE_LVD3	F_AFE_LVD2	0	F_AFE_LVD0	F_LVD_PMC
W				w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_LVD_IO	F_LVD_PLL	F_LVD_FLASH	F_HVD_ADC	F_LVD_ADC	F_HVD_CORE	F_LVD_CORE	F_EDR	0	F_SSSR	F_STCU_SUF	F_FFRR	F_SOFT_DEST	0		F_POR
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

RGM_DES field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 F_LVD_MIPI	LVD on MIPI voltage 0 No MIPI LVD voltage 'destructive' reset event has occurred since the last clear 1 A MIPI LVD voltage 'destructive' reset event has occurred
4 F_AFE_LVD9	**AFE LVD 'destructive' reset event has occurred due to AFE DAC internal capacitor circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
5 F_AFE_LVD6	**AFE LVD 'destructive' reset event has occurred due toAFE SDPLL analog circuitry**

Table continues on the next page...

RGM_DES field descriptions (continued)

Field	Description
	0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
6 F_AFE_LVD8	**AFE LVD 'destructive' reset event has occurred due to level shifter logic** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
7 F_AFE_LVD7	**AFE LVD 'destructive' reset event has occurred due to AFE SDPLL digital circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
8 F_TSR_DEST	**Flag for temperature sensor ‘destructive’ reset** 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
9 F_AFE_LVD5	**AFE LVD 'destructive' reset event has occurred due to AFE OSC circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
10 F_AFE_LVD4	**AFE LVD 'destructive' reset event has occurred due to AFE DAC external capacitor circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
11 F_AFE_LVD3	**AFE LVD 'destructive' reset event has occurred due to AFE ADC digital circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
12 F_AFE_LVD2	**AFE LVD 'destructive' reset event has occurred due to AFE ADC analog circuitry** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 F_AFE_LVD0	**AFE LVD 'destructive' reset event has occurred due to VDD_HV_RAW or VDD_HV_DAC.** 0 No AFE LVD 'destructive' reset event has occurred since the last clear 1 An AFE LVD 'destructive' reset event has occurred
15 F_LVD_PMC	**LVD on PMC voltage** **NOTE:** A destructive reset on the PMC voltage supply LVD (LVD_PMC) will also set the external reset status in RGM_FES[F_EXR]. 0 No PMC LVD voltage 'destructive' reset event has occurred since the last clear 1 A PMC LVD voltage 'destructive' reset event has occurred
16 F_LVD_IO	**LVD on I/O voltage** 0 No I/O LVD voltage 'destructive' reset event has occurred since the last clear 1 An I/O LVD voltage 'destructive' reset event has occurred
17 F_LVD_PLL	**LVD on PLL voltage**

Table continues on the next page...

RGM_DES field descriptions (continued)

Field	Description
	0 No PLL LVD voltage 'destructive' reset event has occurred since the last clear 1 A PLL LVD voltage 'destructive' reset event has occurred
18 F_LVD_FLASH	**LVD on Flash voltage** 0 No Flash LVD voltage 'destructive' reset event has occurred since the last clear 1 A Flash LVD voltage 'destructive' event has occurred
19 F_HVD_ADC	**HVD on ADC voltage** 0 No ADC HVD voltage 'destructive' reset event has occurred since the last clear 1 An ADC HVD voltage 'destructive' reset event has occurred
20 F_LVD_ADC	**LVD on ADC voltage** 0 No ADC LVD voltage 'destructive' reset event has occurred since the last clear 1 An ADC LVD voltage 'destructive' reset event has occurred
21 F_HVD_CORE	**HVD on core voltage** 0 No core HVD voltage 'destructive' reset event has occurred since the last clear 1 A core HVD voltage 'destructive' reset event has occurred
22 F_LVD_CORE	**LVD on core voltage** 0 No core LVD voltage 'destructive' reset event has occurred since the last clear 1 A core LVD voltage 'destructive' reset event has occurred
23 F_EDR	**Flag for ‘functional’ reset escalation** 0 No ‘functional’ reset escalation event has occurred since the last clear 1 A ‘functional’ reset escalation event has occurred
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_SSSR	Flag for SSCM Secure reset 0 No SSCM Secure reset event has occurred since either the last clear or the last power-on reset assertion 1 A SSCM Secure reset event has occurred
26 F_STCU_SUF	**'Destructive' reset event based on STCU offline selftest and STCU Critical Fault (SUF)** **NOTE:** The F_STCU_SUF destructive reset will not occur if the JCOMP is 1 0 No SUF 'destructive' reset event has occurred since last clear 1 A SUF 'destructive' reset event has occurred
27 F_FFRR	**Flag for FCCU failure to react reset** 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
28 F_SOFT_DEST	**Flag for software ‘destructive’ reset** 0 No software ‘destructive’ reset event has occurred since the last clear 1 A software ‘destructive’ reset event has occurred
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

RGM_DES field descriptions (continued)

Field	Description
31 F_POR	**Flag for Power-on reset** 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

62.4.2 'Destructive' Event Reset Disable Register (RGM_DERD)

When a 'destructive' reset source is disabled, the associated 'destructive' event will trigger either a SAFE mode request or an interrupt request (see 'Destructive' Event Alternate Request Register (RGM_DEAR)). This register can be accessed as read/write-only in either supervisor mode or test mode. This register can be accessed as read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on.

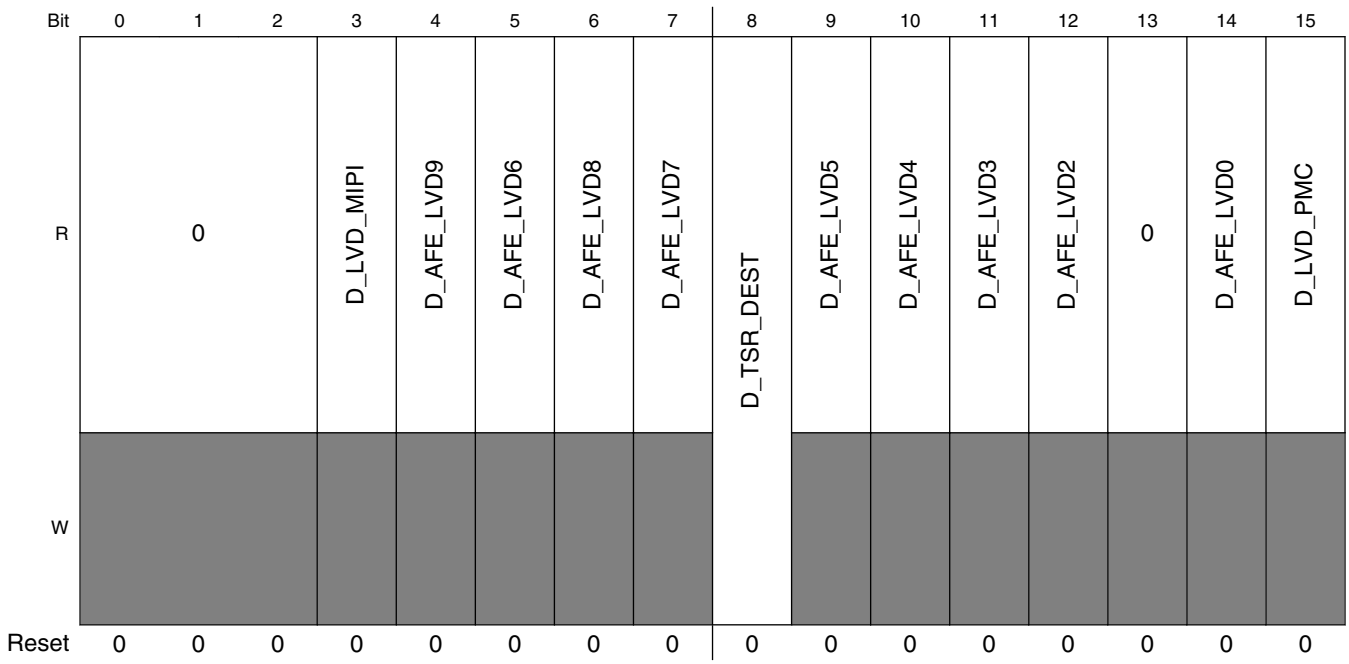
NOTE

If LVD/HVD event is disabled and RGM is configured to issue an interrupt instead of reset, operation below LVD or above HVD levels is not guaranteed. User should guarantee that if LVD/HVD is disabled, then the voltage input is always in range.

CAUTION

It is important to clear the **RGM_DES** register before setting any of the bits in the **RGM_DERD** register to '1'. Otherwise, a redundant ***SAFE*** mode request or interrupt request may occur.

Address: 0h base + 10h offset = 10h



RGM_DERD field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 D_LVD_MIPI	Disable LVD on MIPI voltage 'destructive' reset 0 An LVD on MIPI voltage ‘destructive’ reset event triggers a reset sequence

Table continues on the next page...

RGM_DERD field descriptions (continued)

Field	Description
4 D_AFE_LVD9	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
5 D_AFE_LVD6	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
6 D_AFE_LVD8	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD‘destructive’reset event triggers a reset sequence
7 D_AFE_LVD7	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
8 D_TSR_DEST	**Disable temperature sensor ‘destructive’ reset** 0 A temperature sensor ‘destructive’ reset event triggers a reset sequence 1 A temperature sensor 'destructive' reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_DEAR[AR_TSR_DEST]**
9 D_AFE_LVD5	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
10 D_AFE_LVD4	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
11 D_AFE_LVD3	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
12 D_AFE_LVD2	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 D_AFE_LVD0	**Disable AFE LVD 'destructive' reset** 0 An AFE LVD ‘destructive’ reset event triggers a reset sequence
15 D_LVD_PMC	**Disable LVD on PMC voltage 'destructive' reset** 0 An LVD on PMC voltage ‘destructive’ reset event triggers a reset sequence
16 D_LVD_IO	**Disable LVD on I/O voltage 'destructive' reset** 0 An LVD on I/O voltage ‘destructive’ reset event triggers a reset sequence
17 D_LVD_PLL	**Disable LVD on PLL voltage 'destructive' reset** 0 An LVD on PLL voltage ‘destructive’ reset event triggers a reset sequence
18 D_LVD_FLASH	**Disable LVD on Flash voltage 'destructive' reset** 0 An LVD on Flash voltage ‘destructive’ reset event triggers a reset sequence
19 D_HVD_ADC	**Disable HVD on ADC voltage 'destructive' reset** 0 An HVD on ADC voltage ‘destructive’ reset event triggers a reset sequence

Table continues on the next page...

RGM_DERD field descriptions (continued)

Field	Description
20 D_LVD_ADC	**Disable LVD on ADC voltage 'destructive' reset** 0 An LVD on ADC voltage ‘destructive’ reset event triggers a reset sequence
21 D_HVD_CORE	**Disable HVD on core voltage 'destructive' reset** 0 An HVD on core voltage ‘destructive’ reset event triggers a reset sequence
22 D_LVD_CORE	**Disable LVD on core voltage 'destructive' reset** 0 An LVD on core voltage ‘destructive’ reset event triggers a reset sequence
23 D_EDR	**Disable ‘functional’ reset escalation** 0 A ‘functional’ reset escalation event triggers a reset sequence
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 D_SSSR	Disable LVD on SSCM voltage 'destructive' reset 0 An LVD on SSCM voltage ‘destructive’ reset event triggers a reset sequence
26 D_STCU_SUF	**Disable STCU offline selftest and STCU CF (SUF) 'destructive' reset** 0 An STCU offline selftest and STCU CF (SUF) 'destructive' reset event triggers a reset sequence
27 D_FFRR	**Disable FCCU failure to react reset** 0 A FCCU failure to react reset event triggers a reset sequence
28 D_SOFT_DEST	**Disable software ‘destructive’ reset** 0 A software ‘destructive’ reset triggers a reset sequence
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 D_POR	**Disable Power-on reset** 0 A power-on event triggers a reset sequence

62.4.3 ‘Destructive’ Event Alternate Request Register (RGM_DEAR)

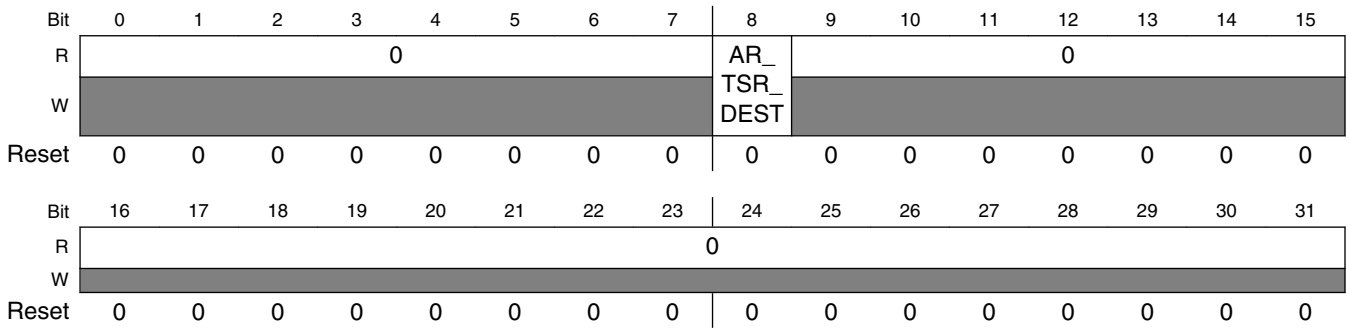
This register defines an alternate request to be generated when a reset on a ‘destructive’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

NOTE

If LVD/HVD event is disabled and RGM is configured to issue an interrupt instead of reset, operation below LVD or above HVD levels is not guaranteed. User should guarantee that if LVD/HVD is disabled, then the voltage input is always in range.

Address: 0h base + 20h offset = 20h



RGM_DEAR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 AR_TSR_DEST	**Alternate Request for temperature sensor 'destructive' reset** 0 Generate a ***SAFE*** mode request on a temperature sensor 'destructive' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'destructive' reset event if the reset is disabled
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.4.4 'Functional' Event Status Register (RGM_FES)

It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source. This register is reset only on power-on.

If functional reset occurs and bidi reset is enabled via RGM_FBRE register with FERD[0] bit set, FES[0] may also get set indicating that external reset event has occurred.

NOTE

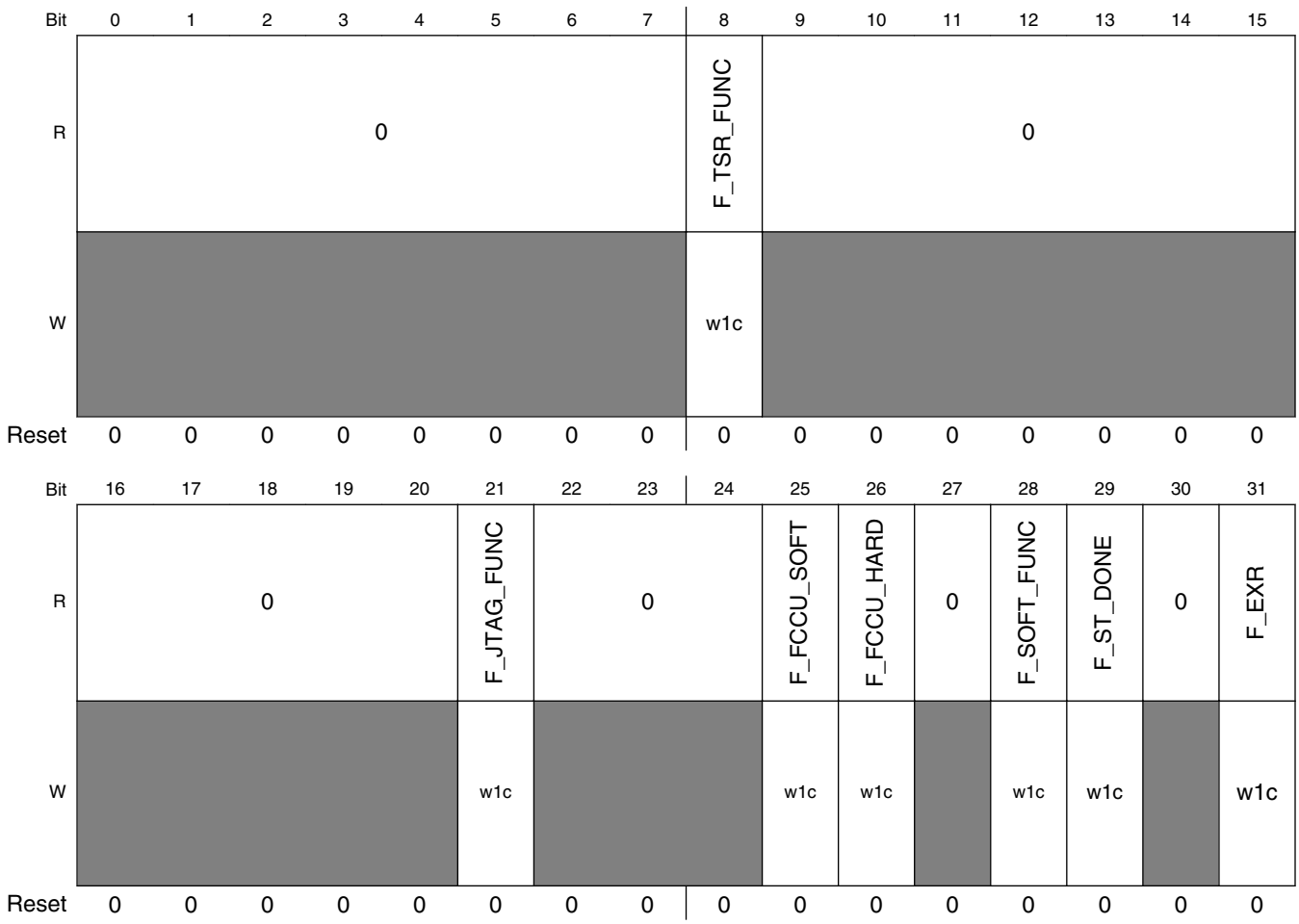
If a 'functional' reset source is configured to generate a ***SAFE*** mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles

before it clears the associated **RGM_FES** status bit in order to avoid multiple ***SAFE mode*** requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the **RGM_FES** has been properly cleared, and if not, clear it again

NOTE

Reset value depends on the source of reset.

Address: 0h base + 300h offset = 300h



RGM_FES field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 F_TSR_FUNC	**Flag for temperature sensor 'functional' reset** 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A temperature sensor 'functional' reset event has occurred.

Table continues on the next page...

RGM_FES field descriptions (continued)

Field	Description
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_JTAG_FUNC	**Flag for JTAG 'functional' reset** 0 No JTAG 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A JTAG 'functional' reset event has occurred.
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_FCCU_SOFT	**Flag for FCCU soft reaction** 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion. 1 A FCCU soft reaction event has occurred.
26 F_FCCU_HARD	**Flag for FCCU hard reaction reset** 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion. 1 A FCCU hard reaction reset event has occurred.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 F_SOFT_FUNC	**Flag for software 'functional' reset** 0 No software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A software 'functional' reset event has occurred.
29 F_ST_DONE	**Flag for self test completed** 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion. 1 A self test completed event has occurred.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_EXR	Flag for extenal reset 0 No external reset has occured since either the last clear or last power-on reset assertion. 1 An external reset event has occured.

62.4.5 'Functional' Event Reset Disable Register (RGM_FERD)

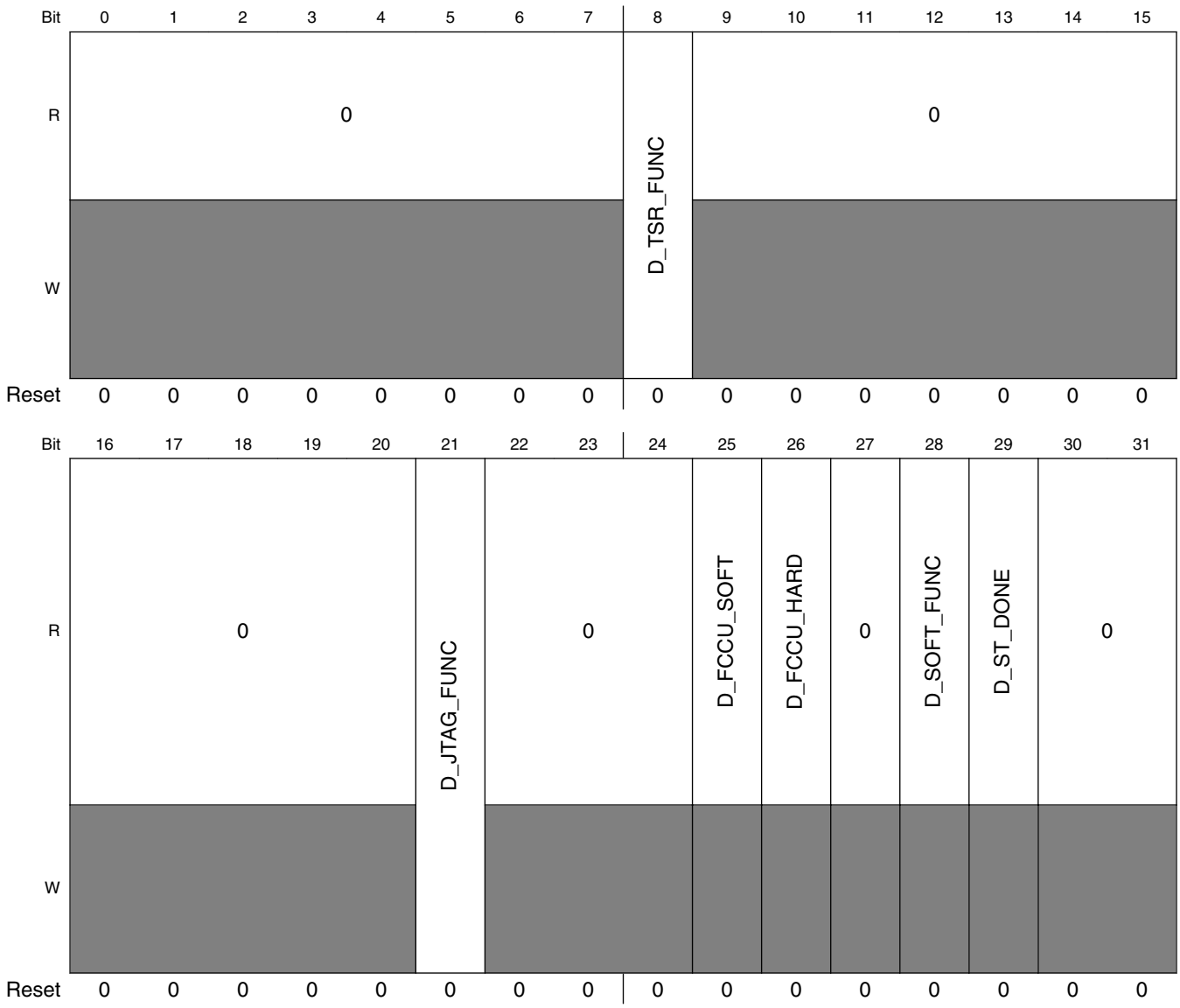
This register provides dedicated bits to disable 'functional' reset sources. When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see 'Functional' Event Alternate Request Register (RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any ‘destructive’ reset.

CAUTION

It is important to clear the **RGM_FES** register before setting any of the bits in the **RGM_FERD** register to ‘1’. Otherwise a redundant ***SAFE*** mode request or interrupt request may occur.

Address: 0h base + 310h offset = 310h



RGM_FERD field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 D_TSR_FUNC	**Disable temperature sensor ‘functional’ reset**

Table continues on the next page...

RGM_FERD field descriptions (continued)

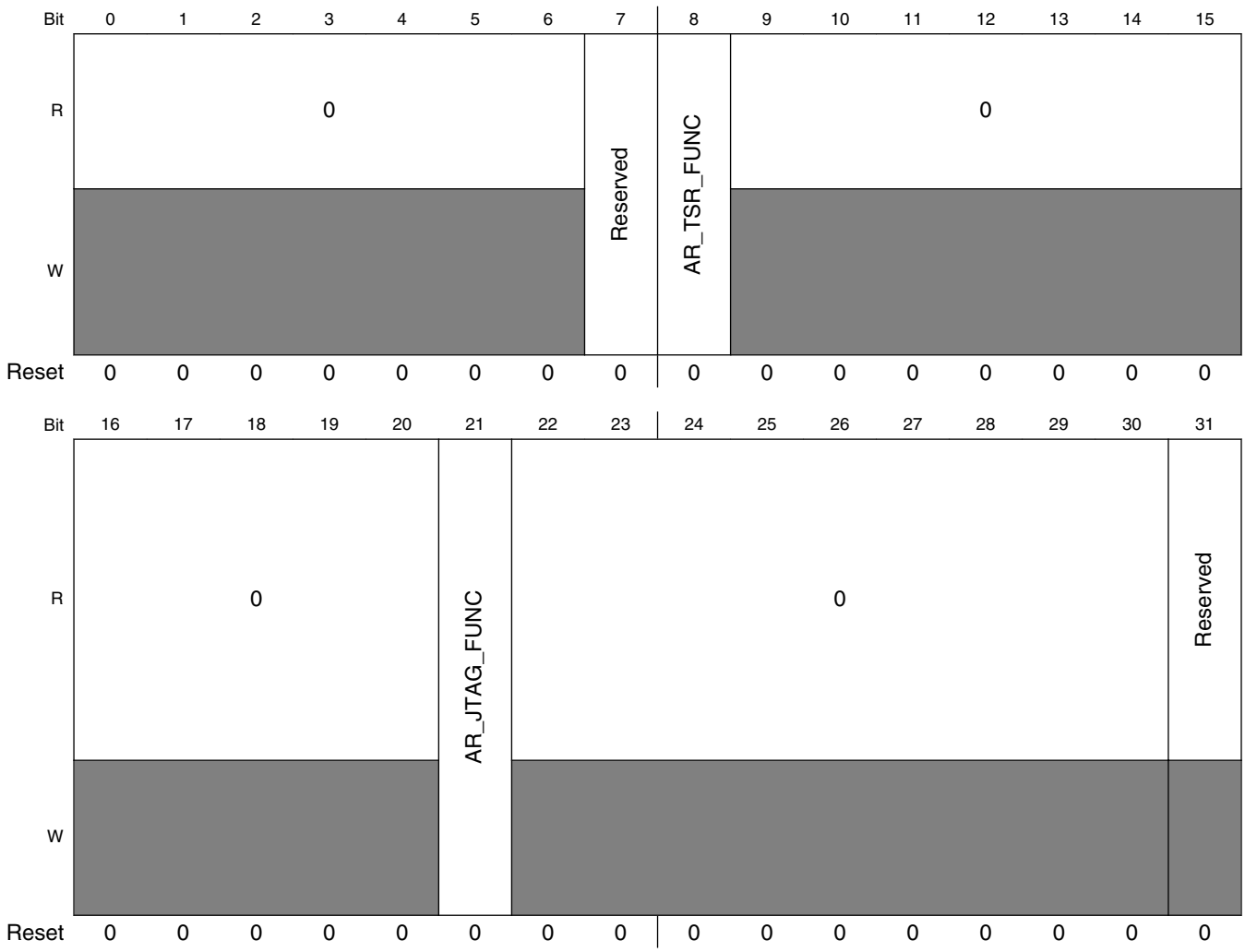
Field	Description
	0 A temperature sensor 'functional' reset event triggers a reset sequence. 1 A temperature sensor 'functional' reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR[AR_JTAG_FUNC]**.
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG_FUNC	**Disable JTAG 'functional' reset** 0 A JTAG 'functional' reset event triggers a reset sequence. 1 A JTAG 'functional' reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR[AR_JTAG_FUNC]**.
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 D_FCCU_SOFT	**Disable FCCU soft reaction** 0 A FCCU soft reaction event triggers a reset sequence.
26 D_FCCU_HARD	**Disable FCCU hard reaction reset** 0 A FCCU hard reaction reset event triggers a reset sequence.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 D_SOFT_FUNC	**Disable software 'functional' reset** 0 A software 'functional' reset event triggers a reset sequence.
29 D_ST_DONE	**Disable self test completed** 0 A self test completed event triggers a reset sequence.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.4.6 'Functional' Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a ***SAFE*** mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 320h offset = 320h



RGM_FEAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.
8 AR_TSR_FUNC	Alternate Request for **temperature sensor 'functional' reset** 0 Generate a ***SAFE*** mode request on a temperature sensor 'functional' reset event if the reset is disabled. 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled.
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AR_JTAG_FUNC	Alternate Request for **JTAG 'functional' reset** 0 Generate a ***SAFE*** mode request on a JTAG 'functional' reset event if the reset is disabled. 1 Generate an interrupt request on a JTAG 'functional' reset event if the reset is disabled

Table continues on the next page...

RGM_FEAR field descriptions (continued)

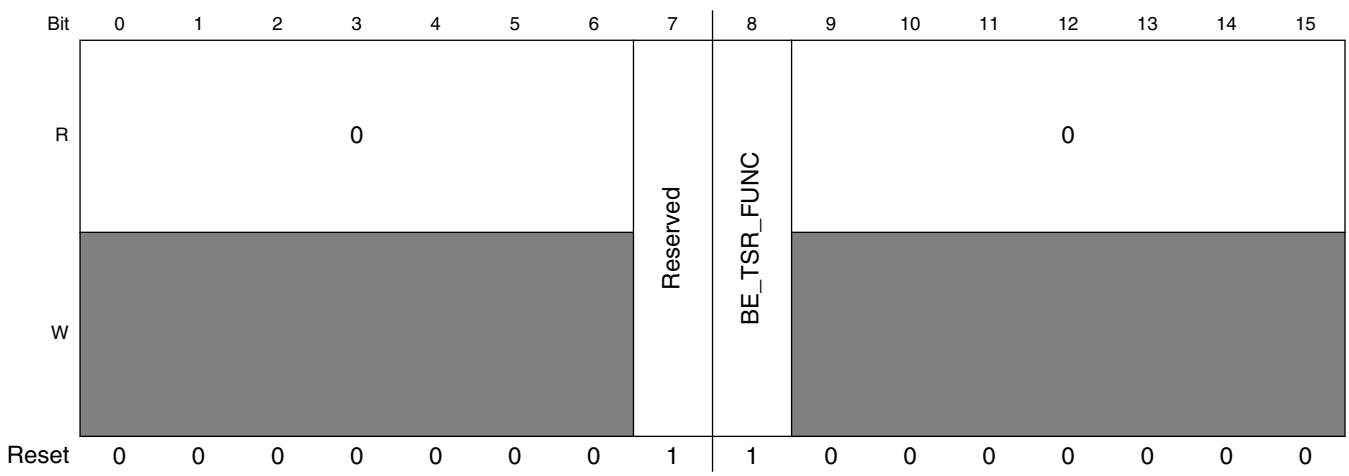
Field	Description
22–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved.

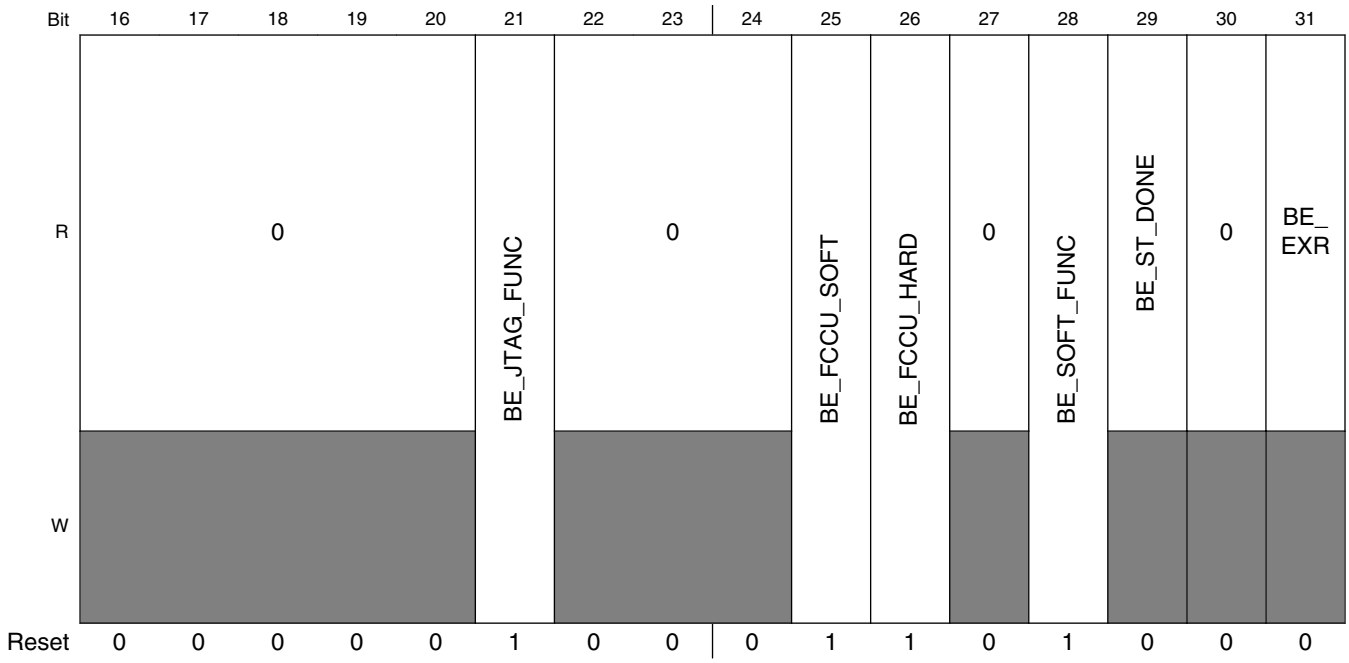
62.4.7 'Functional' Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 330h offset = 330h





RGM_FBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.
8 BE_TSR_FUNC	**Bidirectional Reset Enable for temperature sensor 'functional' reset** 0 RESET_B is asserted on a temperature sensor 'functional' reset event if the reset is enabled. 1 RESET_B is not asserted on a temperature sensor 'functional' reset event.
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BE_JTAG_FUNC	**Bidirectional Reset Enable for JTAG 'functional' reset** 0 RESET_B is asserted on a JTAG 'functional' reset event if the reset is enabled. 1 RESET_B is not asserted on a JTAG 'functional' reset event
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 BE_FCCU_SOFT	**Bidirectional Reset Enable for FCCU soft reaction** 0 RESET_B is asserted on a FCCU soft reaction event. 1 RESET_B is not asserted on a FCCU soft reaction event.
26 BE_FCCU_ HARD	**Bidirectional Reset Enable for a FCCU hard reaction** 0 RESET_B is asserted on a FCCU hard reaction reset event. 1 RESET_B is not asserted on a FCCU hard reaction reset event.

Table continues on the next page...

RGM_FBRE field descriptions (continued)

Field	Description
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BE_SOFT_FUNC	**Bidirectional Reset Enable for software 'functional' reset** 0 RESET_B is asserted on a software 'functional' reset event if the reset is enabled. 1 RESET_B is not asserted on a software 'functional' reset event.
29 BE_ST_DONE	**Bidirectional Reset Enable for self test completed** 0 RESET_B is asserted on a self test completed event.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 BE_EXR	**Bidirectional Reset Enable for external reset** 0 RESET_B is asserted on an external reset event if the reset is enabled.

62.4.8 'Functional' Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from ***PHASE1*** or from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***.

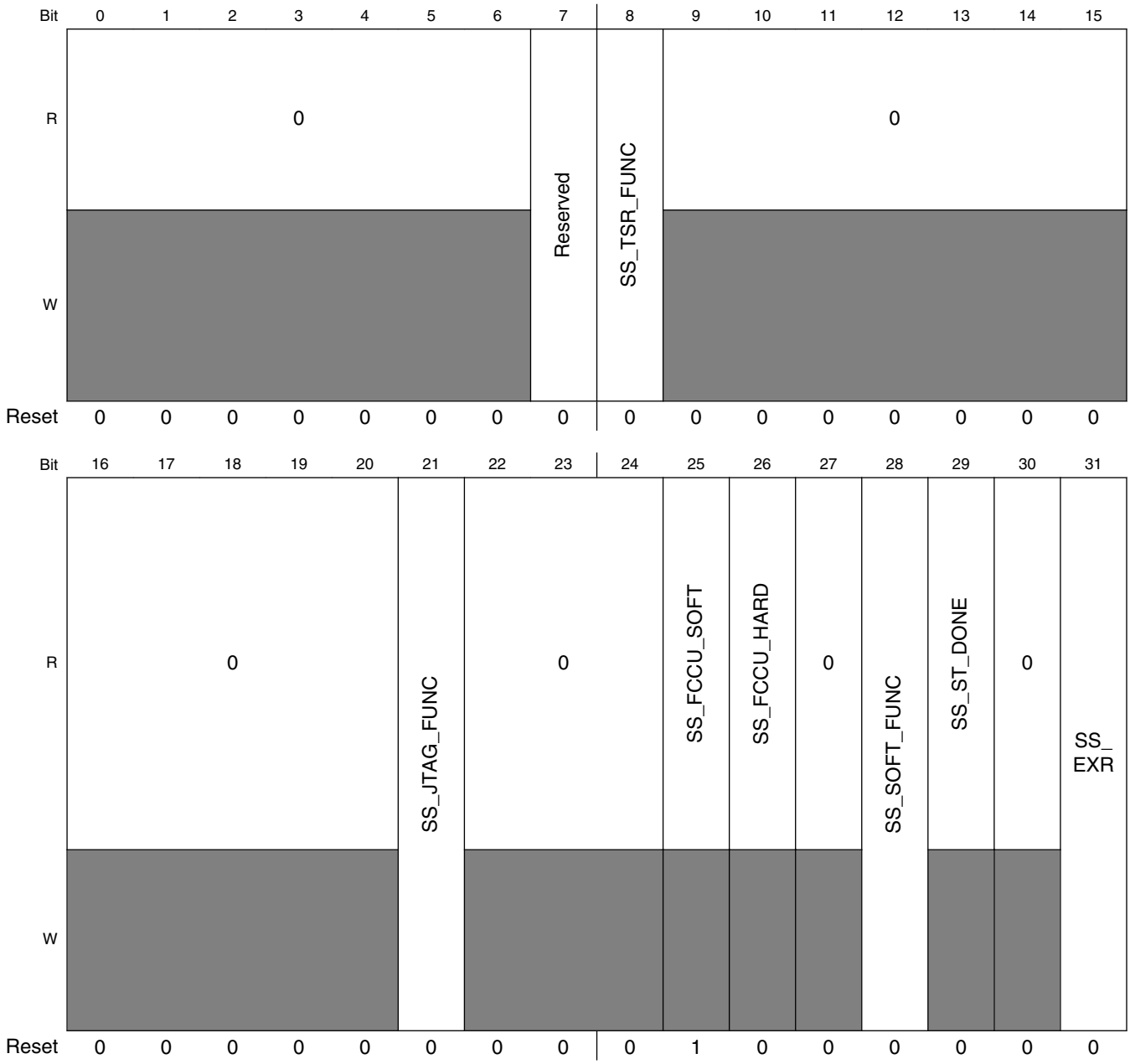
NOTE

This could be useful for fast reset sequence, for example to skip flash reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 340h offset = 340h



RGM_FESS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.
8 SS_TSR_FUNC	**Short Sequence for temperature sensor 'functional' reset** 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from ***PHASE1***. 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***.

Table continues on the next page...

RGM_FESS field descriptions (continued)

Field	Description
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SS_JTAG_FUNC	**Short Sequence for JTAG 'functional' reset** 0 The reset sequence triggered by a JTAG 'functional' reset event will start from ***PHASE1***. 1 The reset sequence triggered by a JTAG 'functional' reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***.
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SS_FCCU_SOFT	**Short Sequence for FCCU soft reaction** 0 The reset sequence triggered by an FCCU Soft reaction event will start from PHASE1
26 SS_FCCU_ HARD	**Short Sequence for a FCCU hard reaction** 0 The reset sequence triggered by an FCCU Hard reaction event will start from PHASE1
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 SS_SOFT_FUNC	**Short Sequence for software 'functional' reset** 0 The reset sequence triggered by a Software functional reset event will start from PHASE1 1 The reset sequence triggered by a Software functional reset event will start from PHASE3, skipping PHASE1 and PHASE2
29 SS_ST_DONE	**Bidirectional Reset Enable for self test completed** 0 RESET_B is asserted on a self test completed event.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SS_EXR	**Short Sequence for external reset** 0 The reset sequence triggered by an external reset event will start from ***PHASE1***. 1 The reset sequence triggered by an external reset event if the reset is enabled will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***.

62.4.9 'Functional' Event Short Sequence Register (RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the **FRET** field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See 'Functional' reset escalation for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7
Read	0				FRET			
Write								
Reset	0	0	0	0	1	1	1	1

RGM_FRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 FRET	**'Functional' Reset Escalation Threshold** — If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets which will cause a 'destructive' reset if the **RGM_FRET** register isn't written to beforehand.

62.4.10 Destructive Reset Escalation Threshold Register (RGM_DRET)

This register sets the threshold for 'destructive' reset escalation to keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.RGM_DRET

Writing a non-zero value to the **DRET** field will enable the 'destructive' reset escalation function. Writing any value to this register will reset the 'destructive' reset counter. See 'Destructive' reset escalation for details on the 'destructive' reset escalation function. escalation function.

This register is reset only on power-on.

Address: 0h base + 608h offset = 608h

Bit	0	1	2	3	4	5	6	7
Read	0				DRET			
Write								
Reset	0	0	0	0	0	0	0	0

RGM_DRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 DRET	**'Destructive' Reset Escalation Threshold** — If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the **>RGM_DRET** register isn't written to beforehand.

62.5 Functional description

62.5.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in the following table.

Table 62-1. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion [1]	Boot Mode Capture
power-on reset	all	yes	yes
'destructive' resets	all except some clock/reset management	yes	yes
external reset	all except some clock/reset management and debug	programmable [2]	yes
'functional' resets	all except some clock/reset management, STCU, FCCU, SSCM and debug	programmable[2]	programmable[3]
shortened external or shortened 'functional' resets[4]	flip-flops except some clock/reset management, STCU, FCCU, SSCM, and debug	programmable[2]	programmable[3]

1. 'external reset assertion' means that the RESET_B pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset ***PHASE3***
2. the assertion of the external reset is controlled via the **RGM_FBRE** register
3. the boot mode is captured if the external reset is asserted
4. the short sequence is enabled via the **RGM_FESS** register

NOTE

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in the following figure.

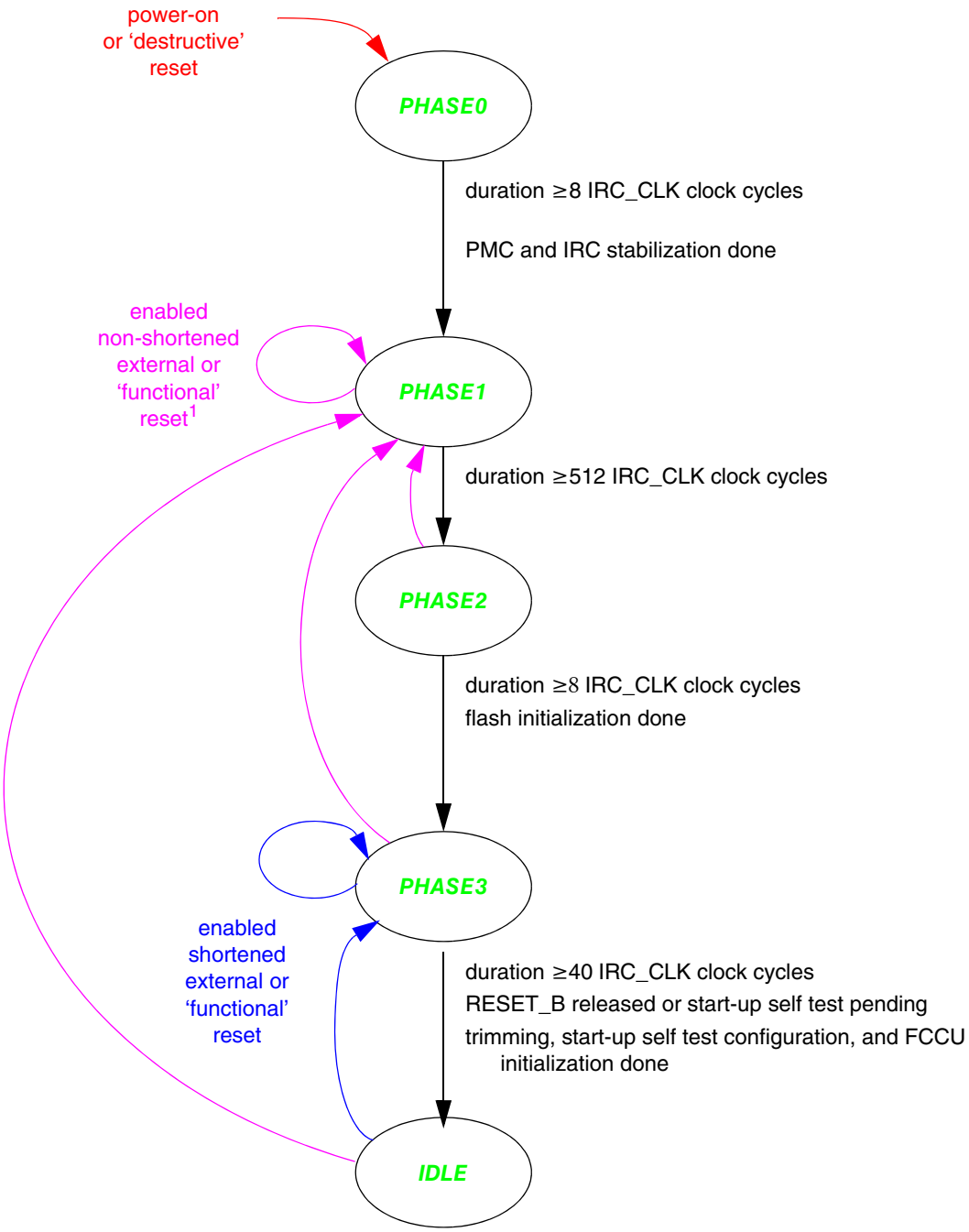


Figure 62-2. MC_RGM State Machine

62.5.1.1 *PHASE0* Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits ***PHASE0*** and enters ***PHASE1*** on verification of the following:

- all enabled 'destructive' resets have been processed

- all processes that need to be done in ***PHASE0*** are completed, including PMC and IRC stabilization
- a minimum of 8 IRC_CLK clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event
- the ‘destructive’ reset escalator counter has not reached the value in the DRET field of the RGM_DRET resister

62.5.1.2 *PHASE1* Phase

This phase is entered either on exit from ***PHASE0*** or immediately from ***PHASE2***, ***PHASE3***, or ***IDLE*** on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits ***PHASE1*** and enters ***PHASE2*** on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of 512 IRC_CLK clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

62.5.1.3 *PHASE2* Phase

This phase is entered on exit from ***PHASE1***. The reset state machine exits ***PHASE2*** and enters ***PHASE3*** on verification of the following:

- all processes that need to be done in ***PHASE2*** are completed flash initialization
- a minimum of 8 IRC_CLK clock cycles have elapsed since entering ***PHASE2***

62.5.1.4 *PHASE3* Phase

This phase is a entered either on exit from ***PHASE2*** or immediately from ***IDLE*** on an enabled, shortened ‘functional’ reset event. The reset state machine exits ***PHASE3*** and enters ***IDLE*** on verification of the following:

- all processes that need to be done in ***PHASE3*** are completed trimming, start-up self test configuration, and FCCU initialization
- a minimum of 40 IRC_CLK clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event

62.5.1.5 *IDLE* Phase

This is the final phase and is entered on exit from ***PHASE3***. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

62.5.2 ‘Destructive’ resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed. Exception to the latter is the software ‘destructive’ reset which does allow the memory content to be preserved if the chip is configured to not execute a self test on power-on, ‘destructive’, and external resets.

The status flag associated with a given ‘destructive’ reset event (**RGM_DES[F_<destructive reset>]** bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘destructive’ reset can be optionally disabled by writing bit **RGM_DERD[D_<destructive reset>]**.

NOTE

The **RGM_DERD** register can be written only once between two ‘destructive’ reset events.

The chip’s low-voltage detector threshold ensures that, when 'destructive' voltage out of range (LVD and/or HVD) is enabled, the supply is sufficient to have the ‘destructive’ event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled ‘destructive’ reset will trigger a reset sequence starting from the beginning of ***PHASE0***.

62.5.3 External reset

The MC_RGM manages the external reset coming from RESET_B. The detection of a falling edge on RESET_B will start the reset sequence.

The status flag associated with the external reset falling edge event (**RGM_FES[F_EXR]** bit) is set when the external reset is asserted and the power-on reset is not asserted.

An enabled external reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by the external reset. When **RGM_FESS[SS_EXR]** is set, the external reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a ‘destructive’ reset event
- an external reset event if configured via the **RGM_FBRE** register to assert the external reset
- a ‘functional’ reset event configured via the **RGM_FBRE** register to assert the external reset

Glitch filter on RESET_B should not be disabled when using it as a bidi pin

In this case, RESET_B is asserted until all conditions for exiting ***PHASE3*** have been met with the exception of the RESET_B assertion check.In addition, the MC_RGM asserts RESET_B while the start-up self test is being executed.

The RESET_B input is disabled immediately as of the RESET_B output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the RESET_B output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

If RESET_B is stopped being asserted by the MC_RGM and the 4 µs input mask period expires during the reset ***IDLE*** phase, and now a low value is detected on RESET_B, an external reset falling edge event will not occur, and no new reset sequence will be triggered. For a new reset sequence to trigger, RESET_B must first be deasserted.

62.5.4 ‘Functional’ resets

A ‘functional’ reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given ‘functional’ reset event (**RGM_FES[F_<functional reset>]** bit) is set when the ‘functional’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘functional’ reset can be optionally disabled by software writing bit **RGM_FERD[D_<functional reset>]**.

NOTE

The **RGM_FERD** register can be written only once between two power-on reset events.

An enabled ‘functional’ reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by a ‘functional’ reset. When **RGM_FESS[SS_<functional reset>]** is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially in case a ‘functional’ reset should not reset the flash module.

62.5.5 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. when a reset source is asserted, the MC_RGM normally enters the reset sequence. alternatively, it is possible for some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the **RGM_D/FERD** and **RGM_D/FEAR** registers as shown in the following table.

Table 62-2. MC_RGM Alternate Event Selection

RGM_D/FERD bit value	RGM_D/FEAR bit value	Generated event
0	X	Reset
1	0	***SAFE*** mode request
1	1	Interrupt request

The alternate event is cleared by deasserting the source of the request (i.e., at the reset source that caused the alternate request) and also clearing the appropriate **RGM_D/FES** status bit.

NOTE

SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

NOTE

If a masked ‘destructive’ reset event which is configured to generate a ***SAFE*** mode/interrupt request occurs during ***PHASE0***, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

NOTE

If a masked ‘functional’ reset event which is configured to generate a ***SAFE*** mode/interrupt request occurs during ***PHASE1***, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

62.5.6 ‘Functional’ reset escalation

‘Functional’ reset escalation can be used to generate a ‘destructive’ reset if a number of ‘functional’ or external resets has occurred between software writes to the **RGM_FRET** register. This function is enabled by writing a non-zero value to the **FRET** field of this register.

Once ‘functional’ reset escalation has been enabled, the MC_RGM increases a counter on each ‘functional’ or external reset which causes a reset sequence to be initiated (i.e., entrance into PHASE1 or ***PHASE3*** from the ***IDLE*** phase). This counter is cleared on a write of any value to the **RGM_FRET** register and on any power-on or ‘destructive’ reset. If the counter reaches the value in the **RGM_FRET** register’s **FRET** field, the MC_RGM starts a reset sequence at ***PHASE0*** and asserts the **F_EDR** bit in the **RGM_DES** register.

62.5.7 ‘Destructive’ Reset Escalation

‘Destructive’ reset escalation can be used to keep the chip in the reset state until the power-on triggers a reset sequence if a number of ‘destructive’ resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once ‘destructive’ reset escalation has been enabled, the MC_RGM increases a counter on each ‘destructive’ reset which causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register’s DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

On a power-on reset, the dest_rst_escalation output is deasserted. It is asserted by the MC_RGM when the counter reaches the value in the RGM_DRET register's DRET field. It is not deasserted until the next power-on reset.

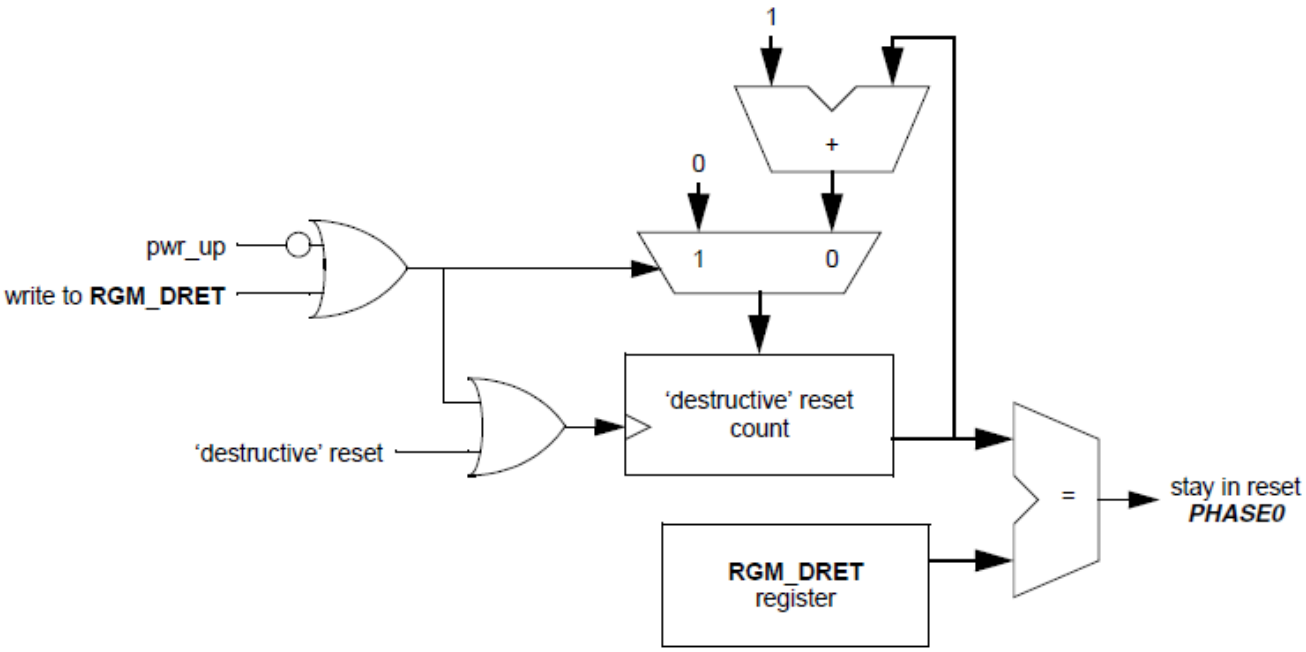


Figure 62-3. MC_RGM 'Destructive ' Reset Escalation to Stay in PHASE0

Chapter 63
System Status and Configuration Module (SSCM)

63.1 Chip-specific SSCM information

63.1.1 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration. There is one instance of the SSCM module on the MCU. For a comprehensive description of the SSCM, please reference the SSCM's dedicated chapter.

The SSCM reads the DCF records during boot, and parses this data to see if valid boot code exists.

SSCM_STATUS[NXEN1] indicates status of core 1(z7a) and core 2 (z7b) nexus enable status.

SSCM_STATUS[NXEN0] indicates status of core 0(z4) nexus enable.

The device specific reset value for the SSCM_MEMCONFIG register are shown in the following table.

Table 63-1. SSCM_MEMCONFIG reset value

Field	Field Definition	Value
JPIN	JTAG Part ID	0011011000b
MREV	Minor Revision	0010b
IVLD	Instruction flash memory valid	1b

Table 63-2 shows the address locations that the SSCM uses to search for startup information.

Table 63-2. SSCM startup information addresses

Description	Address
The location at which the SSCM will look for the UTEST records (UTEST_OFFSET)	0x0040_0300h

The SSCM module has a built-in register protection mechanism that affects the behavior of write operations to one or more registers. After software has activated the protection for a register, writes can only be performed in supervisor mode until the protection is either deactivated by software or the device undergoes reset.

See the Register protection (REG_PROT) configuration for a list of affected SSCM registers.

See the Overview for a detailed description of the register protection mechanism.

63.1.2 SSCM User Option Status Register (SSCM_UOPS)

Address offset: 20h

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BOOT_CORE	0	LOCK_STEP_EN	0											SWT1_EN	SWT0_EN
W																
Reset	*	*	*						*						*	*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SEC_BOOT_CONFIG	SEC_BOOT_EN	0	NVM_CFG											
W																
Reset	*	*	*	*						*						

Note Reset value depends on DCF settings in the Flash or DCF configurations. Refer the DCF sheet attached with this document.

Table 63-3. SSCM UOPS register field descriptions

Field	Description
BOOT_CORE	Indicates if boot core is z4 or Z7a. 0 Z4 is the boot core 1 Z7a is the boot core
LOCK_STEP_EN	Indicates status of lock step mode 0 Disabled 1 Enabled
SWT1_EN	Indicates SWT1 status through flash configuration. 0 Disabled 1 Enabled
SWT0_EN	Indicates SWT0 status through flash configuration. 0 Disabled 1 Enabled
SEC_BOOT_CONFIG	Indicates the selected secure boot mode 0 Concurrent Secure boot mode 1 Sequential secure boot mode
SEC_BOOT_EN	Indicates if secure boot status through flash configuration). 0 Disabled 1 Enabled
NVM_CFG	NVM configuration FCCU_CFG[SM], FCCU_CFG[PS], FCCU_CFG[FOM] and FCCU_CFG[FOP]

63.1.3 ECC Error Monitoring

In case of any ECC error during flash memory scanning (such as Life Cycle , RCHW, or DCF records), the application is able to monitor ECC Errors in the Flash (through EER and ADDR) and can choose an appropriate action.

63.2 Introduction

63.2.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

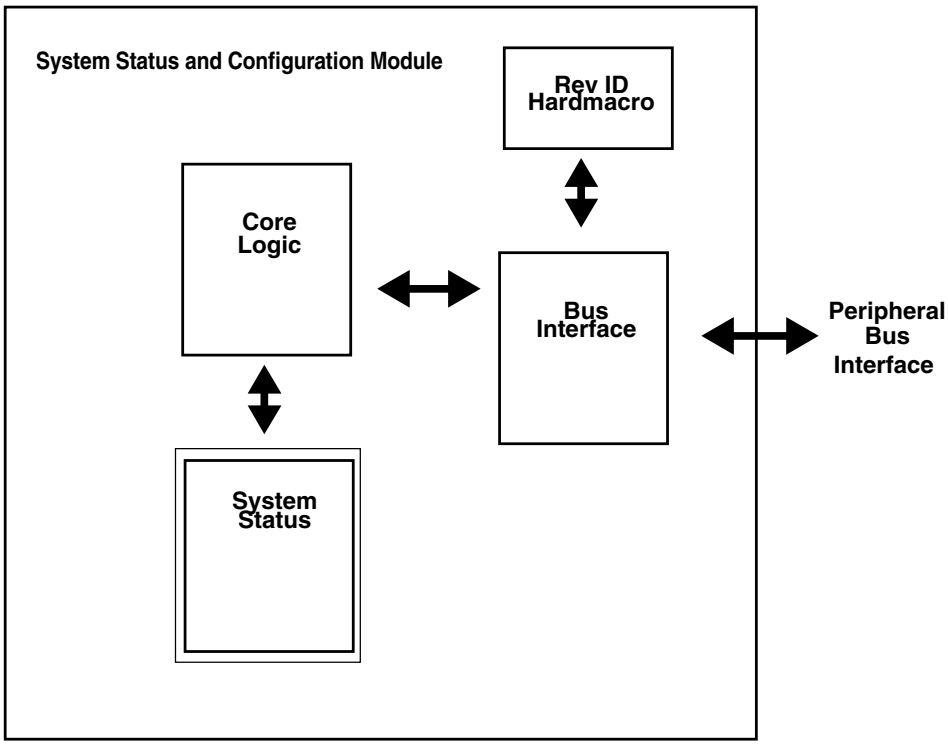


Figure 63-1. SSCM block diagram

63.2.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 63-4. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

63.2.3 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
  - Decodes the Device Life Cycle
- Bus abort enable/disable
- Off-platform Peripheral abort enable/disable

63.2.4 Modes of operation

The SSCM operates identically in all system modes.

63.3 External signal description

The SSCM has no external pins.

63.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal (that is, you should not make any transfer that is not aligned to burst size boundary).

NOTE

SSCM does not generate transfer errors in all memory holes.

NOTE

Reserved IPS register access is prohibited.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	63.4.1/3320
2	SSCM System Memory and ID Register (SSCM_MEMCONFIG)	16	R	See section	63.4.2/3322
6	SSCM Error Configuration Register (SSCM_ERROR)	16	R/W	0000h	63.4.3/3322
28	Processor Start Address Register (SSCM_PSA)	32	R	See section	63.4.4/3323
34	Life Cycle Status Register (SSCM_LCSTAT)	32	R	See section	63.4.5/3324

63.4.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	CERS	NXEN1	NXEN	1	Reserved	0
Write		w1c	w1c					
Reset	0	0	0	*	*	1	*	0

Bit	8	9	10	11	12	13	14	15
Read	BMODE			1	0	0	0	
Write								
Reset	*	*	*	1	0	0	0	0

* Notes:
- BMODE field: Reset value depends on the device status after leaving reset.
- Reserved field: This field can reset to 0 or 1.
- NXEN field: Reset value depends on the device status after leaving reset.
- NXEN1 field: Reset value depends on the device status after leaving reset..

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
2 CERS	Configuration Error for Safe DCF Clients This field indicates that the SSCM has detected a configuration error during reset while loading the Safe DCF Clients (DCF client with SPRD ADDR enabled, Triple Voting Enabled). If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 BMODE	Device Boot Mode This field is only updated during reset. 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single Chip(no Boot flash memory)
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.4.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG part ID.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	JPIN										IVLD	MREV				0
Write																
Reset	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	0

* Notes:
- MREV field: Reset value can be found in the SSCM Configuration section.
- JPIN field: Reset value reflects the JTAG part ID of the device.

SSCM_MEMCONFIG field descriptions

Field	Description
0–9 JPIN	JTAG Part ID Number **NOTE:** Reset value reflects the JTAG part ID of the device.
10 IVLD	Instruction flash memory Valid This bit identifies whether or not the on-chip instruction flash memory is accessible in the system memory map. 0 Instruction Flash is not accessible 1 Instruction Flash is accessible
11–14 MREV	Minor Mask Revision
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.4.3 SSCM Error Configuration Register (SSCM_ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0														PAE	RAE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSCM_ERROR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

14
PAE
	Peripheral Bus Abort Enable.
	
	This bit enables bus aborts on:
	• any access to a peripheral slot that is not used on the device.
	  • This feature is intended to aid in debugging when developing application code.
	• illegal accesses to off-platform peripherals.
	  • On platform peripherals cannot be controlled by SSCM peripheral abort.
	
	0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception
	1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception

15 RAE	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. **NOTE:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the AIPS level). In this case, the PAE and RAE register bits have no effect on the abort. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception

63.4.4 Processor Start Address Register (SSCM_PSA)

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR																															
W																																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- SADR field: Reset value is the start address

SSCM_PSA field descriptions

Field	Description
0–31 SADR	Processor Start Address The boot processor will start executing application code from this address. In SC mode this indicates the location determined by the RCHW search. In the case of a serial download, this register will point to the BAM start location.

63.4.5 Life Cycle Status Register (SSCM_LCSTAT)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		0					LC		
W																
Reset	0	0	0	0	0	0	*	*	0	0	0	0	0	*	*	*

* Notes:
- LC field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- Reserved field: Reset Value is '00' for a valid secured part.

SSCM_LCSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LC	Life Cycle Determined by the SSCM by reading the LC slots. The encoding is: 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

63.5 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

63.5.1 CERS-DCF Mechanism

The figure below shows that a faulty DCF record has been loaded between the original DCF record set and the final DCF record set. When a faulty DCF record is encountered, the SSCM_STATUS[CERS] field indicates that the device configuration is not correct. In a situation like the one shown below, CERS can be cleared by writing 1 to it, as long as the final DCF record set is correct.

If there is an error in the final set of loaded DCF records, CERS cannot be cleared. This situation indicates that the final DCF record set has at least one error. To fix this, the user should update flash memory after the faulty DCF record with a correct DCF record of the same type. For example, if the faulty DCF record is DCF2, then the correct DCF record should also be DCF2.

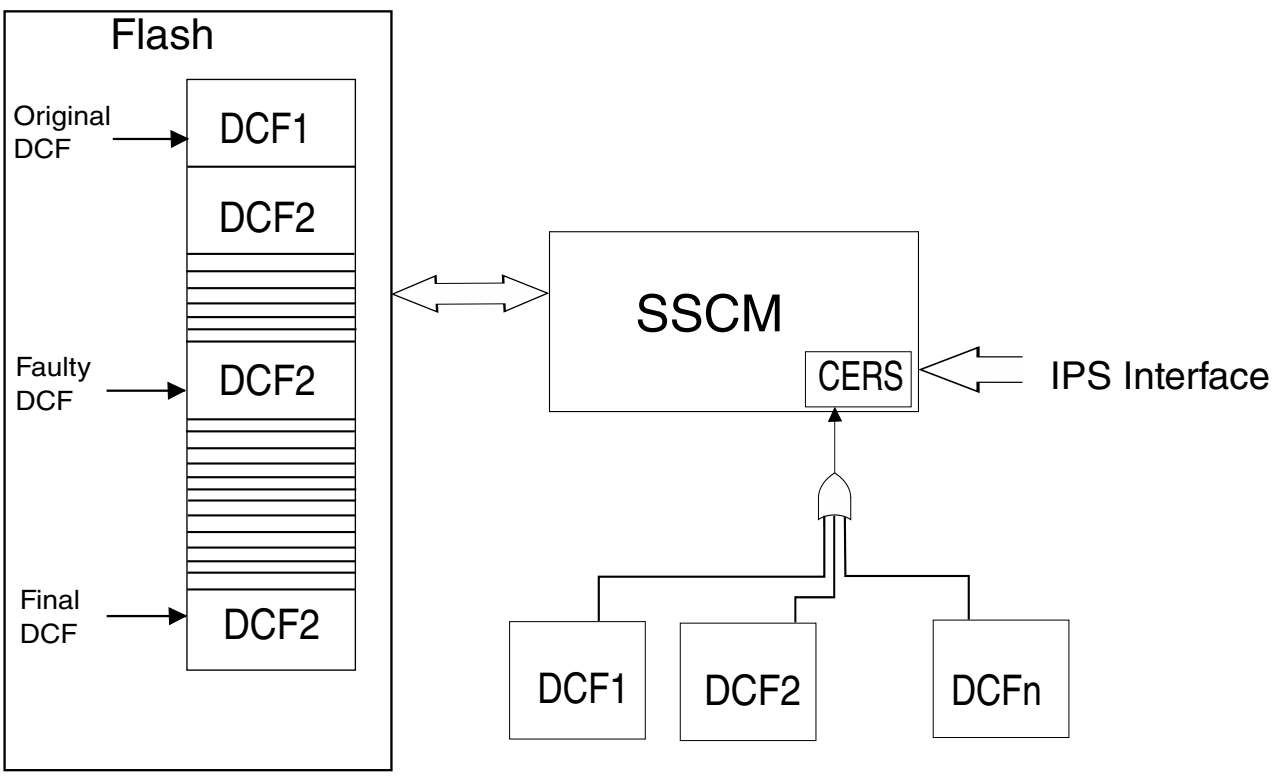


Figure 63-2. DCF Error and CERS Handling

63.5.2 ECC error monitoring

NOTE

During DCF Scanning, if multibit ECC error on DCF start record and DCF start record pattern is corrupted then, SSCM will not be able to identify that as a valid start record and will ignore the flash region as well. In this case, an ECC error address would be logged in . SoC software can monitor for any address of start record. If any start record address found in address log then software should generate a security reset.

Refer to chip-specific SSCM information section for ECC error monitoring and reaction.

63.5.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Single Chip (SC) - The device will boot from the first bootable section of the flash memory main array.
- Serial Boot (SBL) - The device will download boot code from either SCI or CAN interface and then execute it.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter static mode.

63.5.4 Hardware configuration

The device will detect the boot mode based on external pins and device status. The following sequence applies:

- If the FAB (Force Alternate Boot Mode) pin is set to boot in serial mode the device can be forced into an Alternate Boot Loader Mode. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pins. For details of the serial boot modes and hardware configuration please refer to the BAM/System Boot chapter.
- If the device identifies a flash memory sector with a valid boot sector, it will boot from the lowest sector. (See Figure 63-3.)
- If none of the flash memory sectors contains a valid boot signature, the device will go into static mode.

63.5.5 Single Chip boot sector search

63.5.5.1 Potential boot sectors

As shown in Figure 63-3 in single chip mode, several locations are searched for a valid boot ID. The lowest sector that starts with a valid boot ID is used to boot the device.

For the flash memory locations that are searched on this device, see the chip-specific details about boot locations in flash memory.

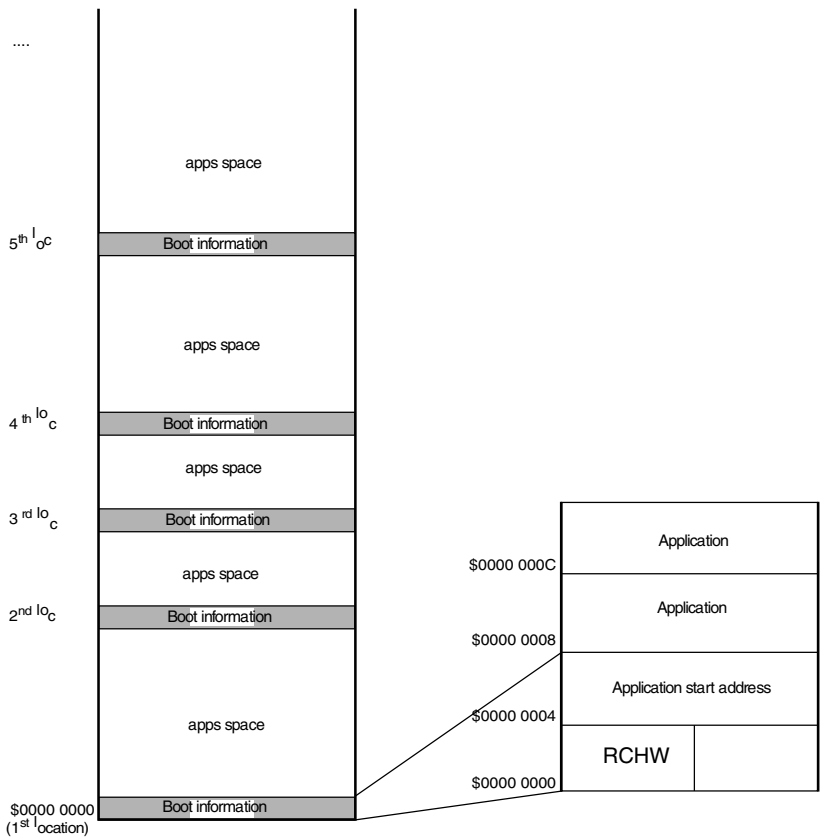


Figure 63-3. Flash memory partitioning and RCHW search

63.5.5.2 Reset Configuration Half-Word

Each boot sector in flash memory contains the Reset Configuration Half-Word (RCHW) at offset 00h. If the RCHW field BOOT_ID holds the value 5Ah then the sector is considered bootable. In addition, there is a flag which indicates that the code is VLE code. All other bits are reserved.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VLE	BOOT_ID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Once the device detects that it needs to boot from flash memory, and finds a valid BOOT_ID, it will boot from the application start address at offset 04h within the boot sector.

NOTE

When programming any of the locations in the RCHW list, ensure that programming is not interrupted and can complete without error. When erasing any block containing any of the locations in the RCHW list, ensure that the erase is not interrupted (especially by reset or loss of power).

63.5.5.3 Boot and alternate boot sectors

Some applications require an alternate boot sector so that the main boot can be erased and reprogrammed in the field.

When an alternate boot is needed, the user can create two bootable sectors; the lowest sector shall be the main boot sector and the highest shall be the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This scheme ensures that there is always one active boot sector by erasing one of the boot sectors only:

- Sector is activated (that is, program a valid BOOT_ID instead of FFh as initially programmed).
- Sector is deactivated by writing 0 to some bits of the BOOT_ID field (bit 1 and/or bit 3, and/or bit 4, and/or bit 6).

63.5.6 Serial boot loader mode

63.5.6.1 Serial boot loader mode - public password enabled

In Serial Boot Loader Mode, if public serial access is allowed, the MCU executes BAM code which will check for a valid public password on the chosen interface. The interface will be selected via FAB and ABS pins.

- JTAG available
- Boot from BAM

63.5.7 Life Cycle

The SSCM will determine the Life Cycle (LC) of the device by reading the LC slots from the UTEST flash memory. The read operation is done during the reset phase with normal timings and it is protected by operating monitors and ECC check. In addition, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- MCU Production
- Customer Delivery
- OEM Production
- In Field
- Failure Analysis

Life cycle can only be moved in the direction shown below:

Figure 63-4. Life cycle sequence

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses as seen in Table 63-5.

Table 63-5. LC slot status

LC slots		LC slot value
Valid field (64 bits)	Invalid field (64 bits)	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		Illegal

"Marked" in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh. "Erased" is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. Table 63-6 shows how the slots are arranged in memory.

Table 63-6. LC slots in memory

Offset	LC slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in Table 63-7

NOTE

"Erased" means the slot returns all 1s.

Table 63-7. LC slots

LC slot 0 MCU Production	LC slot 1 Customer Delivery	LC slot 2 OEM Production	LC slot 3 In Field	LC slot 4 Failure Analysis	Resulting life cycle
Active	Erased	Erased	Erased	Erased	MCU Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	Inactive	**Active**	Erased	Erased	OEM Production
Inactive	Inactive	Inactive	**Active**	Erased	In Field
Inactive	Inactive	Inactive	Inactive	**Active**	Failure Analysis
Erased	Erased	Erased	Erased	Erased	System Reset
Illegal[1]	Illegal	Illegal	Illegal	Erased	In Field

1. Any value programmed other than shown in table above for Erased, Active, and Inactive would be treated as illegal.

NOTE

When programming Life Cycle, ensure that programming is not interrupted and can complete without error.

NOTE

Any other combination of values will result in a system reset.

NOTE

No one should program “ILLEGAL” even if able to do that.
Device behavior is not guaranteed for these combinations.

63.6 Initialization and application information

63.6.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

Chapter 64
Power Management Controller block (PMC)

64.1 Chip-specific Power Management Controller (PMC) information

64.1.1 Overview

The device needs a set of digital and analog supplies and the high fidelity analog supplies. The following figure shows how the different supplies are derived.

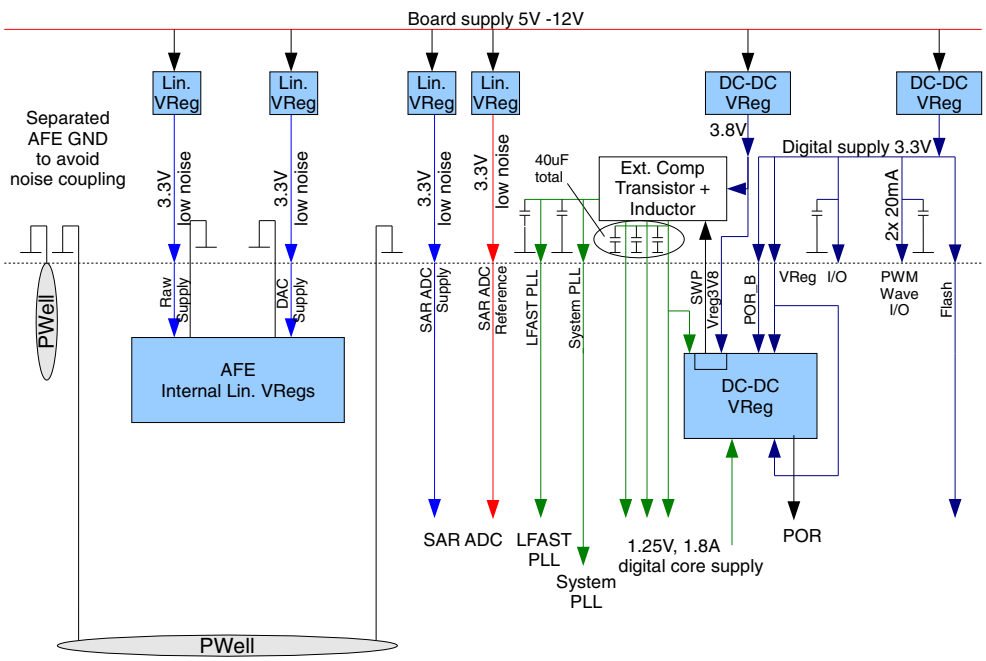


Figure 64-1. Supply Tree

In its standard configuration, the device utilizes the internal DC-DC converter to supply the internal core. This supply is too noisy for supplying the high fidelity analog modules. It might be also required to separate the external 3.3V supplies. For the digital part the HV supply can be derived using a DC-DC converter, while the analog HV supplies shall

be derived from low noise voltage regulators. To avoid spurs in between the high fidelity components via the supply rails, their supplies are separated for the ADC, the clock generation, and the DAC.

In case the high fidelity components are not required to operate in high fidelity mode, the external supply scheme can be simplified to run from a single 3.3V rail.

The digital LV supply at 1.25V can be generated alternatively completely from external. This might make sense in case the external supply is available anyway. In that case the external supply needs to compensate for the internal IR drop.

The following figure shows the case with external core supply and unified 3.3V supply not supporting high fidelity analog processing.

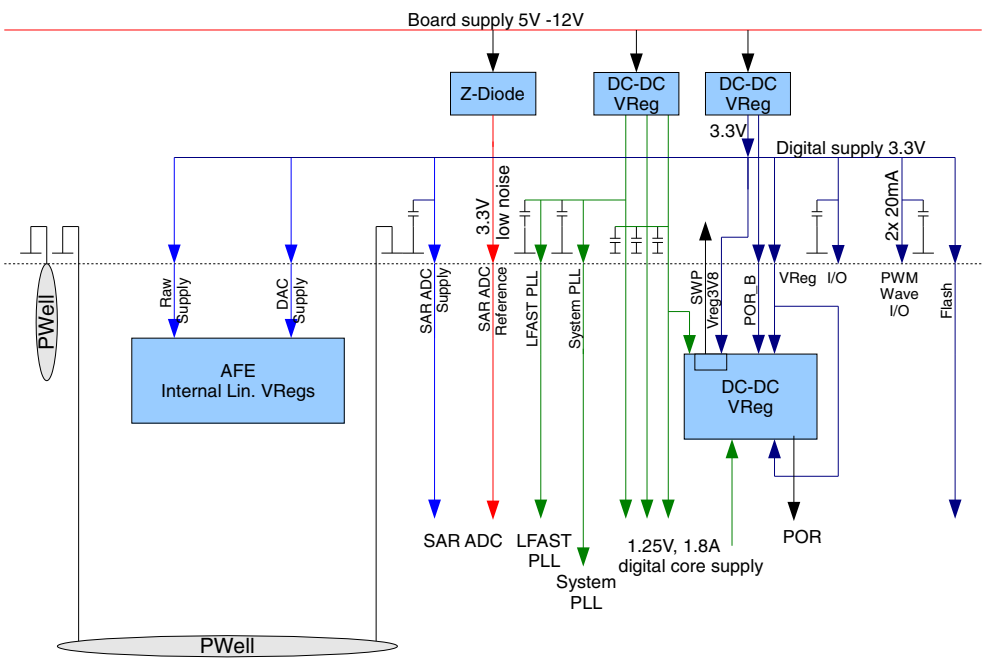


Figure 64-2. Simplified External Supply Tree

When chip is in external regulation mode, the internal POR circuit is disabled. Chip is controlled by POR_B pin. It should be ensured that POR_B should be deasserted only when all supplies are in operating range. LVD and HVD circuit on VDD are disabled by default in external regulation mode. It should be ensured that they are trimmed as per board conditions and only then they are enabled by software or an external LVD/HVD circuit should be used in external regulation.

The following table specifies which LVD/HVD are enabled and disabled in internal and external regulation mode.

Table 64-1. LVD/HVD in internal and external regulation modes

LVD/HVD	Internal Regulation Mode	External Regulation Mode
LVD PMC	Enabled	Enabled
LVD CORE	Enabled	Disabled
HVD CORE	Enabled	Disabled
LVD ADC	Enabled	Enabled
HVD ADC	Enabled	Enabled
LVD Flash	Enabled	Enabled
LVD PLL	Enabled	Enabled
LVD I/O	Enabled	Enabled
LVD MIPI	Disabled	Disabled
1.2V POR	Enabled	Disabled

64.1.2 Power sequencing

Refer device datasheet for power sequencing requirements.

64.1.3 Software-controlled temperature sensor trim adjustment

The SoC provides a feature to update Temperature sensor trims through software. This feature requires an on-chip SAR ADC, this SAR ADC converts the analog voltage provided by temperature sensor into a digital code. The analog voltage provided by temperature sensor is directly proportional to the chip junction temperature. Hence, user should develop a graph of digital ADC code vs. the chip junction temperature reported by corresponding temperature sensor to use this feature. For ADC-TSENS channel details, refer ADC pin muxing.

The following sequence should be used to update temperature sensor trims reliably.

Software sequence for trimming Temperature Sensor 0:

1. Program PMC_REE_TD[TEMP0_0]=0 , PMC_REE_TD[TEMP0_2]=0 to disable reset reaction.
2. Program PMC_TS_IER[TS_EN]=0 to disable temperature sensor interrupt reaction.
3. Program FCCU_NCFS_CFG0[NCFSCx[1:0]] = 00 , to disable FCCU fault reaction.
4. Program PMC_CTL_TD [TS0_DOUT_EN] =0, PMC_CTL_TD[TS0_AOUT_EN]=0 , to disable temperature sensor.
5. Program PMC_CTL_TD [TS0_TRIM_ADJ [4:0]] to update temperature sensor trims to desired value.
6. Wait for 25 us to allow temperature sensor stabilization with new trim values.

7. Program PMC_CTL_TD [TS0_DOUT_EN] =1 , PMC_CTL_TD[TS0_AOUT_EN]=1 , to enable temperature sensor.
8. Check the temperature by performing ADC conversion on Temperature sensor0 ADC channel which is connected to channel no.15 on ADC0.
9. If the temperature is in range (-40°C to 150°C):
   - Clear PMC_ESR_TD[TEMP0_0] and PMC_ESR_TD [TEMP0_2].
   - Program PMC_REE_TD/PMC_TS_IER/ FCCU_NCFS_CFG0 to allow resets, interrupts or fault configuration, respectively.
10. If the temperature is not in range, perform the necessary safety actions for the application to bring the temperature within range.

Software sequence for trimming Temperature Sensor 1:

1. Program PMC_REE_TD[TEMP1_0]=0 , PMC_REE_TD[TEMP1_2]=0 to disable reset reaction.
2. Program PMC_TS_IER[TS_EN]=0 to disable temperature sensor interrupt reaction.
3. Program FCCU_NCFS_CFG0[NCFSCx[1:0]] = 00, to disable FCCU fault reaction.
4. Program PMC_CTL_TD [TS1_DOUT_EN] =0, PMC_CTL_TD[TS1_AOUT_EN]=0, to disable temperature sensor.
5. Program PMC_CTL_TD [TS1_TRIM_ADJ [4:0]] to update temperature sensor trims to desired value.
6. Wait for 25 us to allow temperature sensor stabilization with new trim values.
7. Program PMC_CTL_TD [TS1_DOUT_EN] =1 , PMC_CTL_TD[TS1_AOUT_EN]=1, to enable temperature sensor.
8. Check the temperature by performing ADC conversion on Temperature sensor1 ADC channel which is connected to channel no.15 on ADC1.
9. If the temperature is in range (-40°C to 150°C):
   - Clear PMC_ESR_TD[TEMP1_0] and PMC_ESR_TD [TEMP1_2].
   - Program PMC_REE_TD/PMC_TS_IER/ FCCU_NCFS_CFG0 to allow resets, interrupts or fault configuration, respectively.
10. If the temperature is not in range, perform the necessary safety actions for the application to bring the temperature within range.

NOTE

The temperature sensor flags in PMC_ESR_TD register may get set when ADC supply is varied below LVD level and its LVD destructive reset is masked.

64.1.4 Enabling FCCU faults from PMC

By default, FCCU faults from the PMC block is masked by the DCF. To enable FCCU faults from PMC, dcl_pmc_ree_ctrl dcf client bit 0 must be set to 1. See Device Configuration Format (DCF) Records chapter for general information on DCF records. See the attached spreadsheet for details on the DCF client.

64.2 PMC introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information. Also refer to the Hardware Design guide for requirements and recommendations on the external design components.

The power management controller (PMC) consists of two blocks: analog block and a digital block.

The PMC block, shown below, contains all of the registers and digital logic to generate all of the enable signals, trim control bits, power on resets (PORs), and test mode logic for the analog block. The PMC digital logic also controls the PMC analog outputs that are to be sensed by the Analog-to-Digital Converter (ADC) module.

The PMC digital logic also provides the enable signals, trim bits and other registers for the temperature sensors.

Custom interfaces are provided to each of:

- The Reset Generation Module (MC_RGM) block, which receives low voltage detect (LVD) values that are used for phase transition conditions during POR.
- The Flash memory, via the SSCM (and DCF client blocks), provides trim values for use during initial POR.

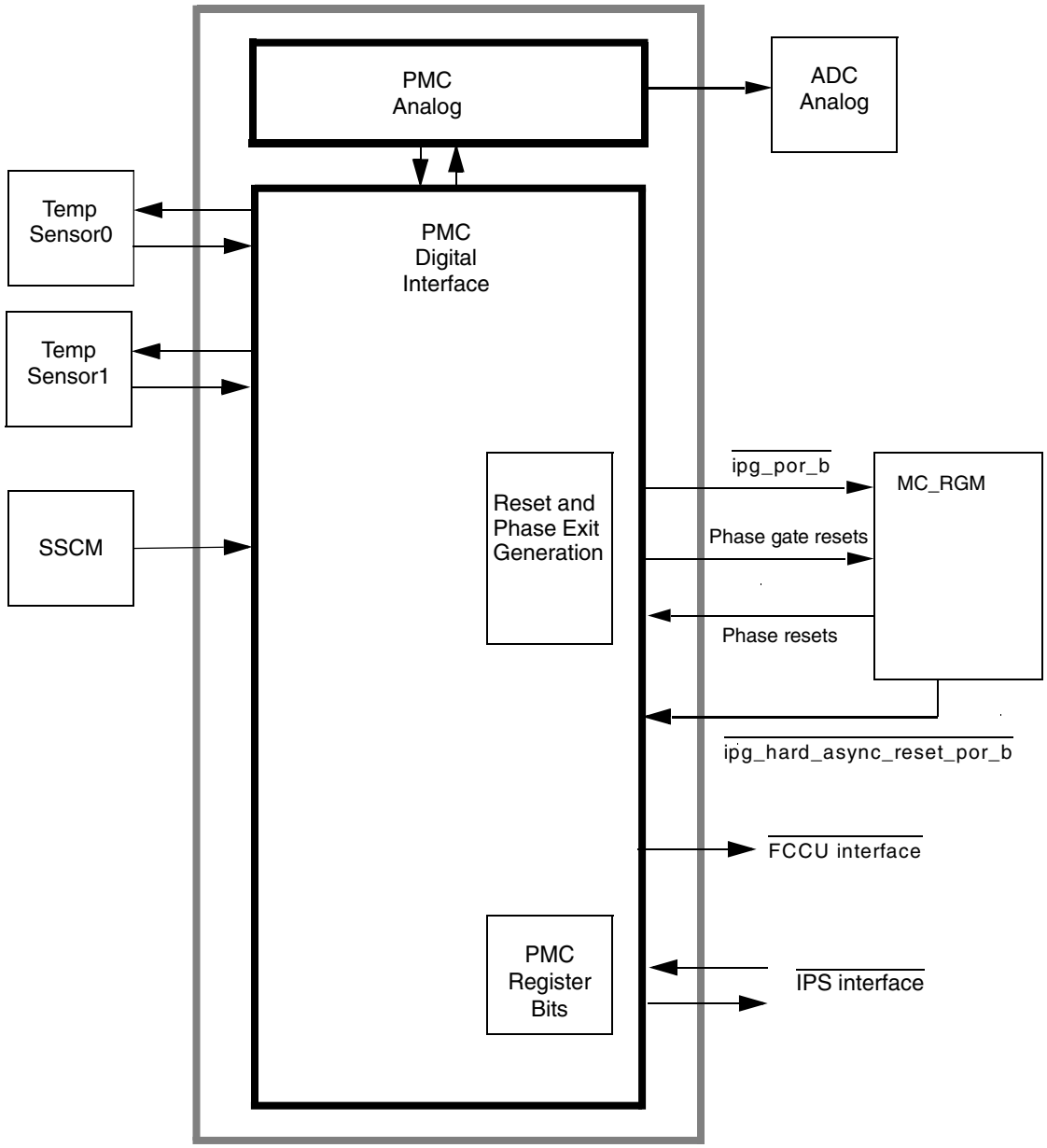


Figure 64-3. Digital PMC block diagram

64.2.1 Features

The PMC module provides:

- Configuration and control information to the analog PMC block.
- Configuration and monitoring of the two temperature sensors.
- Host access, via the IPS bus, to all register controls, configurations, and status.
- Self test logic for the LVDs and HVDs.
- Transfer wait state feature supported on IPS bus access.
- Compatible with 3.3V (analog) operation.

- 1.25 V / 1.8 A output voltage switch-mode power supply (SMPS) asynchronous buck regulator.
- Precision LVD (Low-voltage detection) and HVD (high-voltage detection) for:
  - PMC voltage supply (VDDREG) - LVD
  - VDD core voltage supply - POR/LVD/HVD
  - VDD ADC voltage supply - LVD/HVD
  - Flash memory voltage supply - LVD
  - Oscillators/PLL voltage supply - LVD
  - VDDIO I/O voltage supply - LVD
- Two independent voltage references provide fault tolerance:
  - Voltage monitors and ADC references
  - Core voltage regulator
- DCF Client interface: An interface with Flash memory for retrieval of trim/control information, during POR for:
  - HVDs/ LVDs Regulators
  - Band gap filters
- MC_RGM Interface: Necessary status and indication information to the MC_RGM block during power up.
- FCCU (Fault control and correction Unit): Software Triggered fault injection for LVDs, HVDs, temperature sensors, and LVD self test.

64.3 Memory map and registers

Please note that 8/16 bit register accesses are not supported.

PMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
64	SMPS Control Register (PMC_SMPS_CNTRL)	32	R/W	See section	64.3.1/3340
68	LVD Self Test Time Window Register (PMC_STTW)	32	R/W	See section	64.3.2/3342
6C	Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)	32	R/W	See section	64.3.3/3343
78	AFE Interrupt Enable Register (PMC_AFE_INTR_ENA)	32	R/W	See section	64.3.4/3345
80	AFE Interrupt Status Register (PMC_AFE_INTR_STATUS)	32	w1c	See section	64.3.5/3347
8C	FCCU Fault Injection Register (PMC_FIR)	32	W	0000_0000h	64.3.6/3349
94	PMC Control Register (PMC_PMCCR)	32	R/W	See section	64.3.7/3350
98	Interrupt Enable Register (PMC_TS_IER)	32	R/W	0000_0000h	64.3.8/3353
9C	Temperature Event Status register (PMC_ESR_TD)	32	R/W	0000_0000h	64.3.9/3355
A0	Temperature Reset Event Enable register (PMC_REE_TD)	32	R/W	See section	64.3.10/ 3357

Table continues on the next page...

PMC memory map (continued)

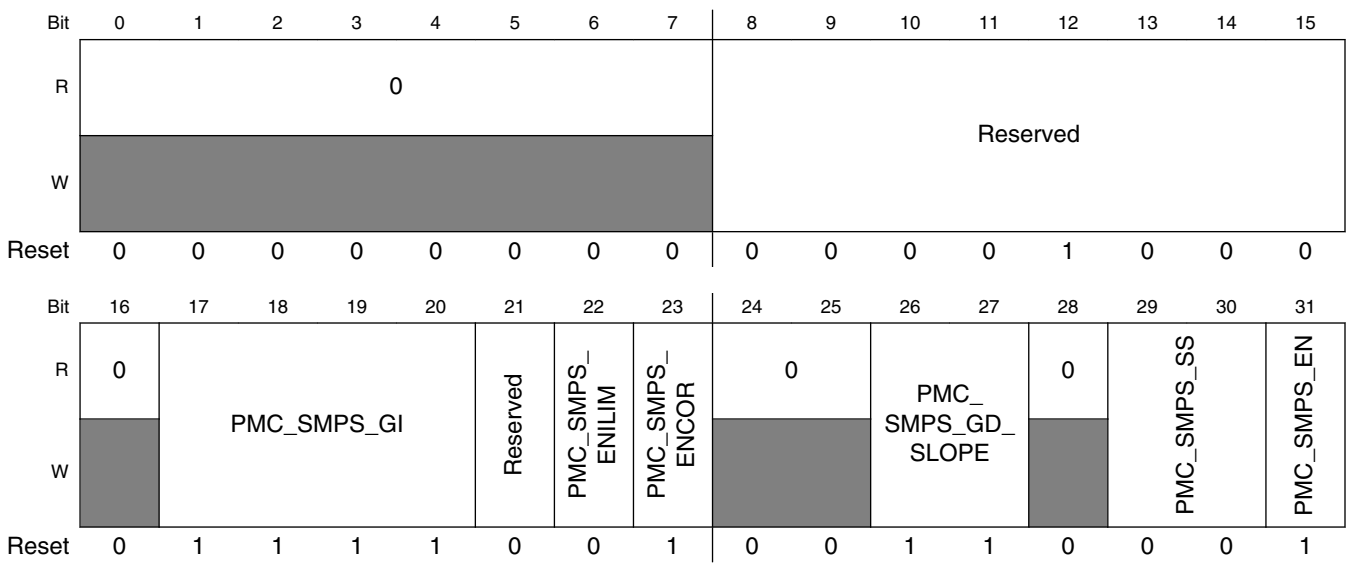
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Temperature Reset Event Selection register (PMC_RES_TD)	32	R/W	See section	64.3.11/ 3358
A8	Temperature detector configuration register (PMC_CTL_TD)	32	R/W	See section	64.3.12/ 3359
B4	Fault Injection Register (PMC_TS_FIR)	32	W	0000_0000h	64.3.13/ 3361

64.3.1 SMPS Control Register (PMC_SMPS_CNTRL)

This register contains the various control settings for the SMPS module.

RESET_B does not reset this register. Only a power-on reset (POR) does.

Address: 0h base + 64h offset = 64h



PMC_SMPS_CNTRL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 Reserved	This field is reserved. These bits should always be set to 8'h08.
16 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
17–20 PMC_SMPS_GI	SMPS Overshoot Current Trimming

Table continues on the next page...

PMC_SMPS_CNTRL field descriptions (continued)

Field	Description
	These trimming bits control the SMPS current limitation feature, when PMC_SMPS_ENLIM = 1. The recommended value is 0110. Please refer to the Hardware Design Guide for more details. This feature is based off of the estimation of the SMPS external inductor current from the RC network connected to vreg_iSENS pin. The RC network is configured such that for a given inductor, the voltage difference between core VDD and vreg_iSENS pin provides an estimate of the inductor current. These trimming bits control the voltage threshold, which in turn selects the max current allowed through the inductor. A 0000 trim setting represents 100mV drop across the intrinsic resistor and each additional count represents an additional 25mV voltage increment. The higher the count of the trim bits, the higher the max current allowed. For a particular external inductor and its associated intrinsic resistance; a particular trim setting is recommended based on current limit required. When the inductor current exceeds the trimmed limit value, the SMPS current limitation loop turns off the external pMOS transistor temporarily. DC loop continues core voltage regulation once inductor current falls within limit.
21 Reserved	This field is reserved. This bit should be programmed to 0.
22 PMC_SMPS_ENILIM	SMPS Inductor Current Limitation Enable This bit enables current limitation feature through the external inductor based on feedback through the ISENS pin. See PMC_SMPS_GI.
23 PMC_SMPS_ENCOR	SMPS Hysteretic Correction Enable ENCOR bit must be turned high after power-up, to limit voltage disturbances during extreme load current transients. It is recommended that the Inductor Current Limitation feature(PMC_SMPS_ENLIM) must be enabled at all times when hysteretic correction is used. 0 Disabled 1 Enabled
24–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PMC_SMPS_GD_SLOPE	SMPS gate driver strength Controls strength of gate-driver circuit. After power-up minimum strength is selected. By adjusting this field, it is possible to trade efficiency for EMI. Stronger driver will generate more EMI. Recommended settings: 01. 00 100% drive strength 01 Approximately 70% drive strength 10 Approximately 50% drive strength 11 Approximately 35% drive strength
28 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
29–30 PMC_SMPS_SS	SMPS spread spectrum control SMPS frequency modulation control. Allow controller to reduce EMI generated at the switching frequencies and its harmonics. Default settings: 00.

Table continues on the next page...

PMC_SMPS_CNTRL field descriptions (continued)

Field	Description
	00 No modulation 01 15% modulation 10 7.5% modulation 11 30% modulation
31 PMC_SMPS_EN	SMPS controller enable In internal regulation mode, the SMPS controller is enabled by default. When an external regulator is to be used, the external SMPS components must not be populated on the board. In external regulation case, this bit can be de-asserted to disable the SMPS controller after power-up. This will allow the use of the internal power-on-reset even with external regulator. 0 SMPS control is disabled 1 SMPS control is enabled

64.3.2 LVD Self Test Time Window Register (PMC_STTW)

This register configures the Self test Time window for each LVD/HVD.

NOTE

The reset value of STTW register depends upon the value of the dcl_pmc_self_test_time_gap DCF Client. Please refer to Device Configuration Format (DCF) Records for further details.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																STTW															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0	1	0	1	0	0	0	0	0

* Notes:
- If DCF record value is programmed, then reset value will be from DCF, else it will be 0xA0.

PMC_STTW field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–31 STTW	Self Test Time Window. These bits are used to control the Time window for the self test for each LVD/HVD. This is based on the clock frequency used. For example, for 21 µs window, with 45 MHz clock, this should be configured with hex value 3E8. **NOTE:** STTW needs to be programmed to attain a window of 21 µs.

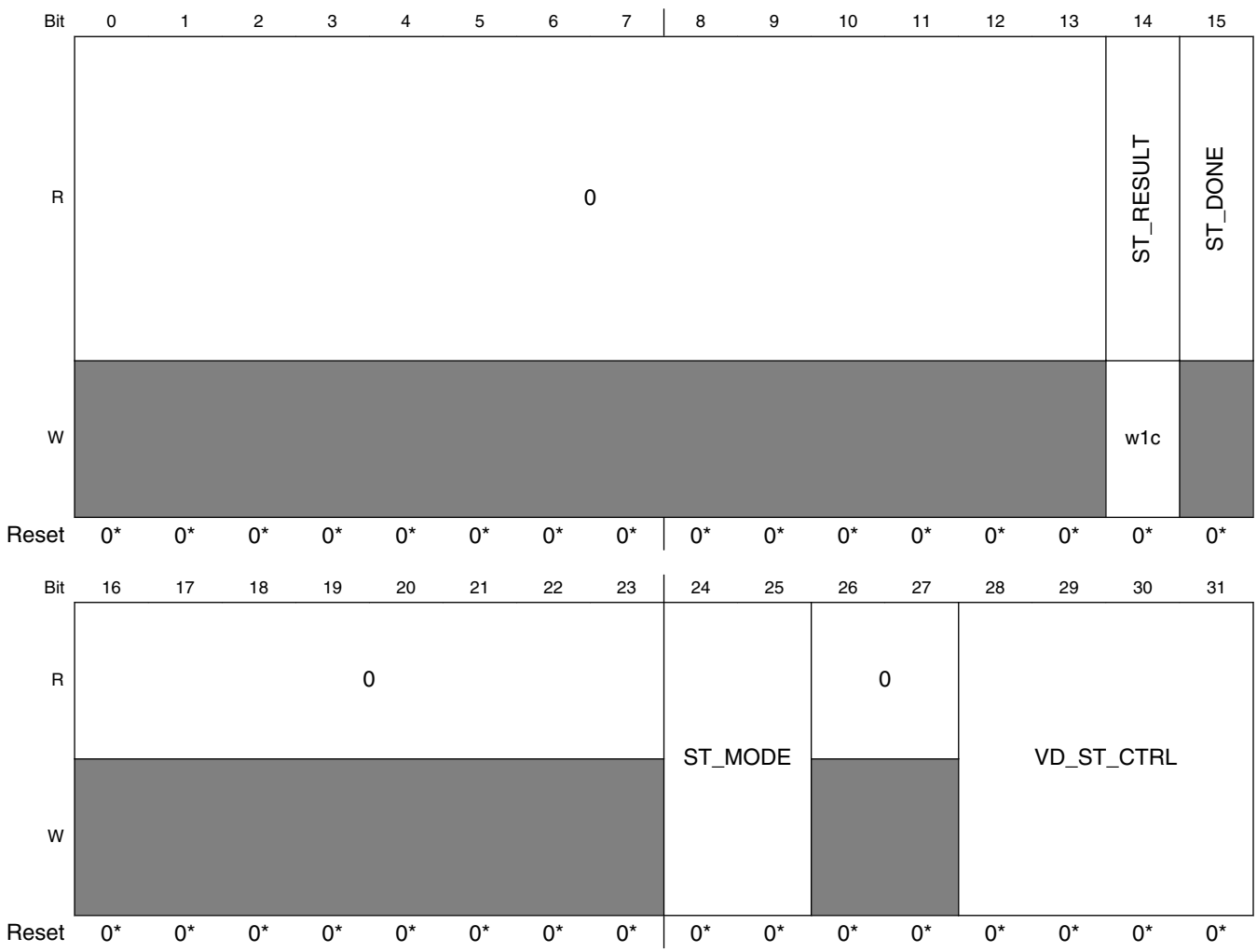
64.3.3 Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)

This register is used for testing of the LVDs and HVDs during user mode. To clear ST_RESULT field after single VD self test, user should ensure that the ST_CTRL field is also written with 0x0 value along with ST_RESULT = 1 (w1c field).

NOTE

This register is reset on a destructive reset. IPS writes on any register should not be done during running of software self test which is indicated by ST_DONE = 0b0.

Address: 0h base + 6Ch offset = 6Ch



* Notes:

- By default SELF TEST is enabled and by the time the reset value is read, SELF TEST results would be reflected.

PMC_SELF_TEST_UM_VD_REG field descriptions

Field	Description
0–13 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
14 ST_RESULT	ST_RESULT. Self Test result status. ST_RESULT. Self Test result status. (Valid only when ST_DONE is High). It is cleared on reset or writing '1' by the host. 0 Self Test Failed 1 Self Test Passed
15 ST_DONE	Self Test Done status. Self Test completed indication status. **NOTE:** There should be no configuration updates till the time ST_DONE bit is set. 0 Self test running 1 Self test completed.
16–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
24–25 ST_MODE	Self Test mode bits for testing of LVDs and HVDs. Mode 01 (software triggered self test) is similar to the default mode, except the user programs the ST_MODE field to start the test. In this mode, all LVD/HVD test modes are executed automatically. Mode 10 (single VD test) executes each test individually. The user sets the test by programming the PMC_SELF_TEST_UM_REG[VD_ST_CTRL] field. If either the default mode or the software triggered self test fail, the user can run the single VD test to determine which LVD/HVD passed and which LVD/HVD failed. 00 Default 01 Software triggered self test. 10 Single VD test as indicated by the VD_ST_CTRL bits. 11 Reserved
26–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28–31 VD_ST_CTRL	Voltage Detect Self Test Control User mode Test bits for testing of LVDs and HVDs. 0000 Functional Mode 0001 Reserved 0010 Core LVD 0011 Core HVD 0100 LVD MIPI 0101 LVD PMC 0110 LVD FLash 0111 LVD ADC

Table continues on the next page...

PMC_SELF_TEST_UM_VD_REG field descriptions (continued)

Field	Description
	1000 LVD PLL 1001 LVD IO 1010 HVD ADC 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

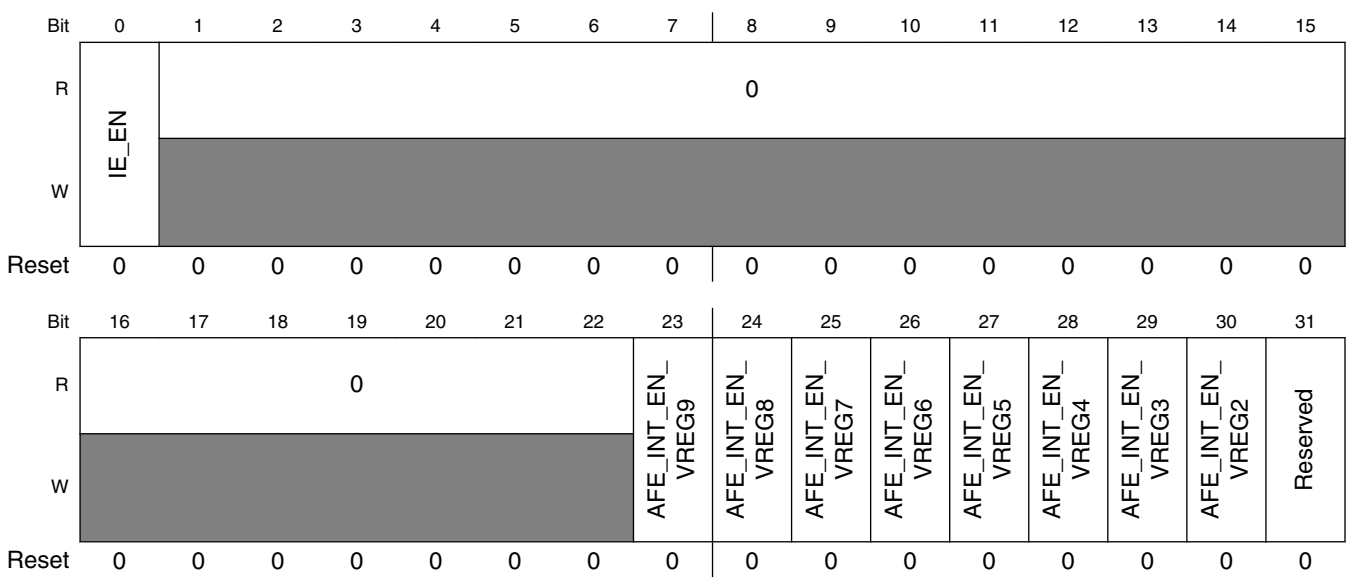
64.3.4 AFE Interrupt Enable Register (PMC_AFE_INTR_ENA)

This configuration register contains a set of AFE Interrupt Enable bits that correspond to each of the AFE Event Status Registers (ESR_0). These bits indicate whether an interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt is to be sent, and a '1' indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.Please refer to AFE VREGn register descriptions for details.

NOTE

This register is reset on a short functional reset.

Address: 0h base + 78h offset = 78h



PMC_AFE_INTR_ENA field descriptions

Field	Description
0 IE_EN	Interrupt Enable. This bit determines whether any of the Interrupt Enable bits can be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
1–22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
23 AFE_INT_EN_ VREG9	AFE-9 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
24 AFE_INT_EN_ VREG8	AFE-8 Interrupt Enable/Disable Control: This field determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
25 AFE_INT_EN_ VREG7	AFE-7 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
26 AFE_INT_EN_ VREG6	AFE-6 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
27 AFE_INT_EN_ VREG5	AFE-5 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
28 AFE_INT_EN_ VREG4	AFE-4 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
29 AFE_INT_EN_ VREG3	AFE-3 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system. 0 No interrupt occurs. 1 An interrupt occurs.
30 AFE_INT_EN_ VREG2	AFE-2 Interrupt Enable/Disable Control This bit determines whether an interrupt is seen by the system.

Table continues on the next page...

PMC_AFE_INTR_ENA field descriptions (continued)

Field	Description
	0 No interrupt occurs. 1 An interrupt occurs.
31 Reserved	This field is reserved.

64.3.5 AFE Interrupt Status Register (PMC_AFE_INTR_STATUS)

This configuration register contains a set of Interrupt Status bits that correspond to each of the AFE Interrupt Event Status. These bits indicate whether an AFE interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt event is generated and a '1' indicates that an interrupt has occurred when the voltage detect event occurs. After this, these bits are W1c type

NOTE

Please refer to AFE VREGn register descriptions for details.

NOTE

The value of this register after reset sequences other than power-on-reset (POR) is not deterministic. The value on the status alone do not indicate an LVD has occurred. To ascertain an LVD occurrence, use the corresponding bit in PMC_AFE_INTR_ENA to enable the LVD as an interrupt or MCB_AFE_LVD_MASK to enable the LVD occurrence as a reset source.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							AFE_INT_STATUS_VREG9	AFE_INT_STATUS_VREG8	AFE_INT_STATUS_VREG7	AFE_INT_STATUS_VREG6	AFE_INT_STATUS_VREG5	AFE_INT_STATUS_VREG4	AFE_INT_STATUS_VREG3	AFE_INT_STATUS_VREG2	0
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value at POR

PMC_AFE_INTR_STATUS field descriptions

Field	Description
0–22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
23 AFE_INT_STATUS_VREG9	AFE9-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. 0 No Interrupt Occurrence has been detected. 1 An Interrupt occurrence has been detected.
24 AFE_INT_STATUS_VREG8	AFE8-Interrupt Status: This bit indicates whether an Interrupt is seen by the system from AFE.
25 AFE_INT_STATUS_VREG7	AFE7-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. 0 No Interrupt Occurrence has been detected. 1 An Interrupt occurrence has been detected.
26 AFE_INT_STATUS_VREG6	AFE6-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. 0 No Interrupt Occurrence has been detected. 1 An Interrupt occurrence has been detected.
27 AFE_INT_STATUS_VREG5	AFE5-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. 0 No Interrupt Occurrence has been detected. 1 An Interrupt occurrence has been detected.
28 AFE_INT_STATUS_VREG4	AFE4-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. Reserved

Table continues on the next page...

PMC_AFE_INTR_STATUS field descriptions (continued)

Field	Description
	0 No Interrupt Occurrence has been detected. 1 An Interrupt occurrence has been detected.
29 AFE_INT_ STATUS_VREG3	AFE3-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. No Interrupt Occurrence has been detected. An Interrupt occurrence has been detected.
30 AFE_INT_ STATUS_VREG2	AFE2-Interrupt Status This bit indicates whether an Interrupt is seen by the system from AFE. No Interrupt Occurrence has been detected. An Interrupt occurrence has been detected.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

64.3.6 FCCU Fault Injection Register (PMC_FIR)

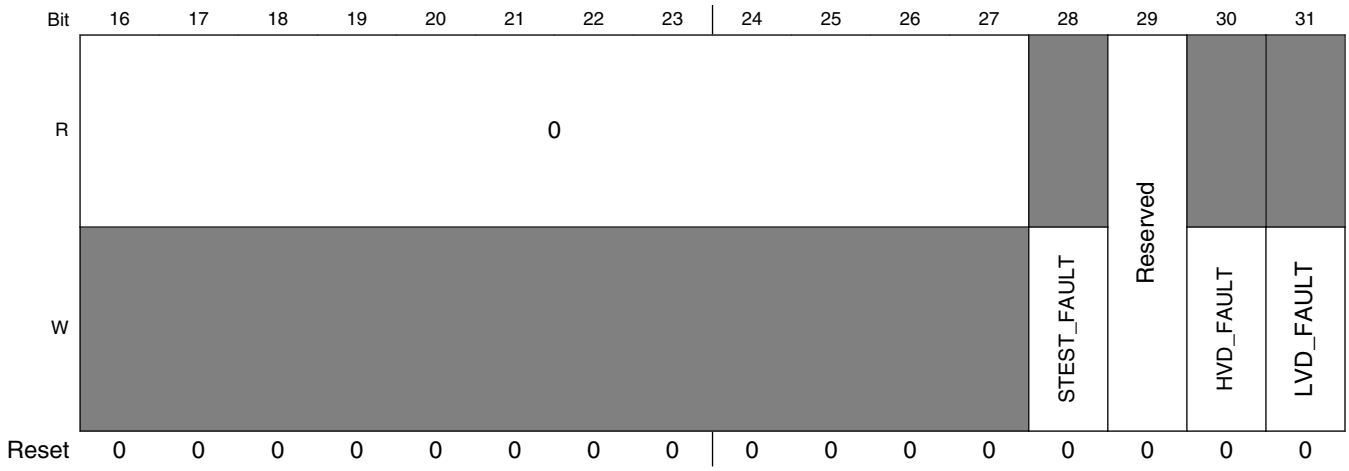
This FCCU Fault Injection register enables the user s/w to artificially generate FCCU faults corresponding to various Events, namely LVD, HVD, and Self Test. A write '1' to these bits generates a single cycle pulsed fault to the FCCU block, this bit automatically gets cleared in the next cycle.

NOTE

This register is reset on a destructive reset.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PMC_FIR field descriptions

Field	Description
0–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 STEST_FAULT	A write to this bit sends a FCCU fault (single cycle) for LVD SELF TEST. This is self Clearing bit. 0 No fault is sent 1 Single cycle fault injected.
29 Reserved	Reserved. This field is reserved.
30 HVD_FAULT	High Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for HVD. This is self clearing bit. 0 No fault is sent 1 Single cycle fault injected.
31 LVD_FAULT	Low Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for LVD. This is self clearing bit. 0 No fault is sent to FCCU 1 Single cycle (one clock cycle) fault is injected to FCCU.

64.3.7 PMC Control Register (PMC_PMCCR)

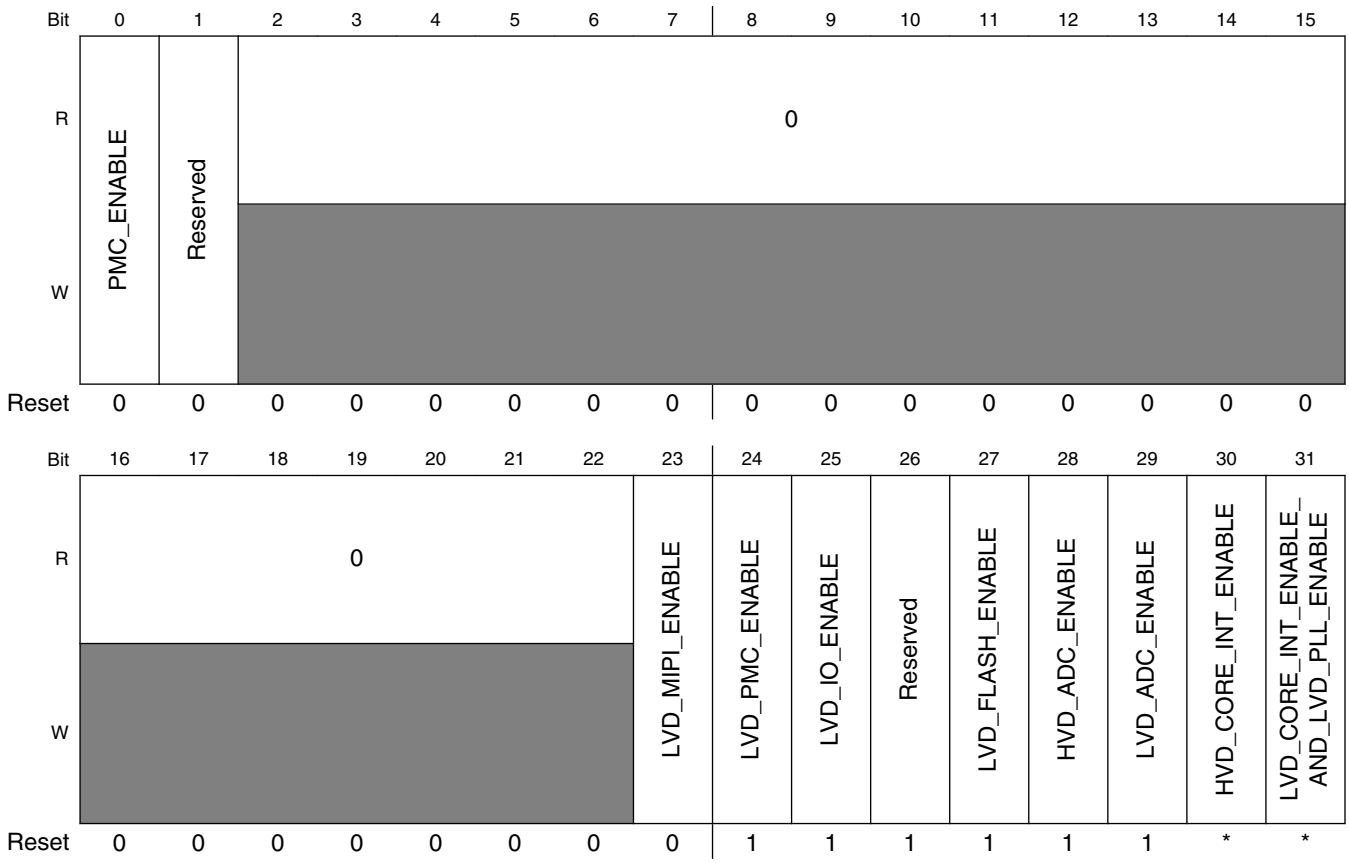
This register contains the various control bits of PMC block.

NOTE

Bits [31:23] of this register gets reset on POR. Bits [1:0] of this register gets reset on destructive reset. Core LVD/HVD are disabled out of reset in external regulation mode. So the value

of bits [31:30] is logic high (1’b1) in internal regulation mode and logic low (1’b0) in external regulation mode out of reset.

Address: 0h base + 94h offset = 94h



* Notes:
- LVD_CORE_INT_ENABLE_AND_LVD_PLL_ENABLE field: See the NOTE above.
- HVD_CORE_INT_ENABLE field: See the NOTE above.

PMC_PMCCR field descriptions

Field	Description
0 PMC_ENABLE	PMC Enable Control This bit determines whether any of the PMC bits can be written or not. 0 No control bit can be written in this(PMC enable control) register. To do a write on this register, first enable this bit and then start writing on other control bits. 1 This Control register can be written.
1 Reserved	Reserved This field is reserved.
2–22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_PMCCR field descriptions (continued)

Field	Description
23 LVD_MIPI_ ENABLE	This bit controls the enable/disable on the LVD MIPI supply. 0 LVD_MIPI is disabled. 1 LVD_MIPI is enabled.
24 LVD_PMC_ ENABLE	This bit controls the enable/disable on the LVD PMC supply. 0 LVD_PMC is disabled 1 LVD_PMC is Enabled
25 LVD_IO_ ENABLE	This bit controls the enable/disable on the LVD IO supply. 0 LVD_IO is disabled 1 LVD_IO is Enabled
26 Reserved	This field is reserved.
27 LVD_FLASH_ ENABLE	This bit controls the enable/disable on the LVD FLASH supply. 0 LVD_FLASH is disabled 1 LVD_FLASH is Enabled
28 HVD_ADC_ ENABLE	This bit controls the enable/disable on the HVD ADC supply. 0 HVD_ADC is disabled 1 HVD_ADC is Enabled
29 LVD_ADC_ ENABLE	This bit controls the enable/disable on the LVD ADC supply. 0 LVD_ADC is disabled 1 LVD_ADC is Enabled
30 HVD_CORE_ INT_ENABLE	This bit controls the enable/disable on the HVD Core supply. This bit resets to 0 in external regulation mode and it resets to 1 in internal regulation mode. 0 HVD_CORE is disabled 1 HVD_CORE is Enabled
31 LVD_CORE_ INT_ENABLE_ AND_LVD_PLL_ ENABLE	LVD Core and LVD_PLL Enable Disable control This single bit does the control of LVD_CORE and LVD _PLL. This bit controls the enable/disable on the LVD Core supply and LVD_PLL as well. This bit resets to 0 in external regulation mode and it resets to 1 in internal regulation mode. **NOTE:** In other words, "LVD_CORE_INT_ENABLE" and "LVD_PLL_ENABLE" control are exactly the same. 0 This Single bit does the control of LVD_CORE and LVD _PLL LVD_CORE_INT_ENABLE is disabled. LVD_PLL_ENABLE is disabled. 1 LVD CORE_INT_ENABLE is Enabled. LVD_PLL_ENABLE is Enabled.

64.3.8 Interrupt Enable Register (PMC_TS_IER)

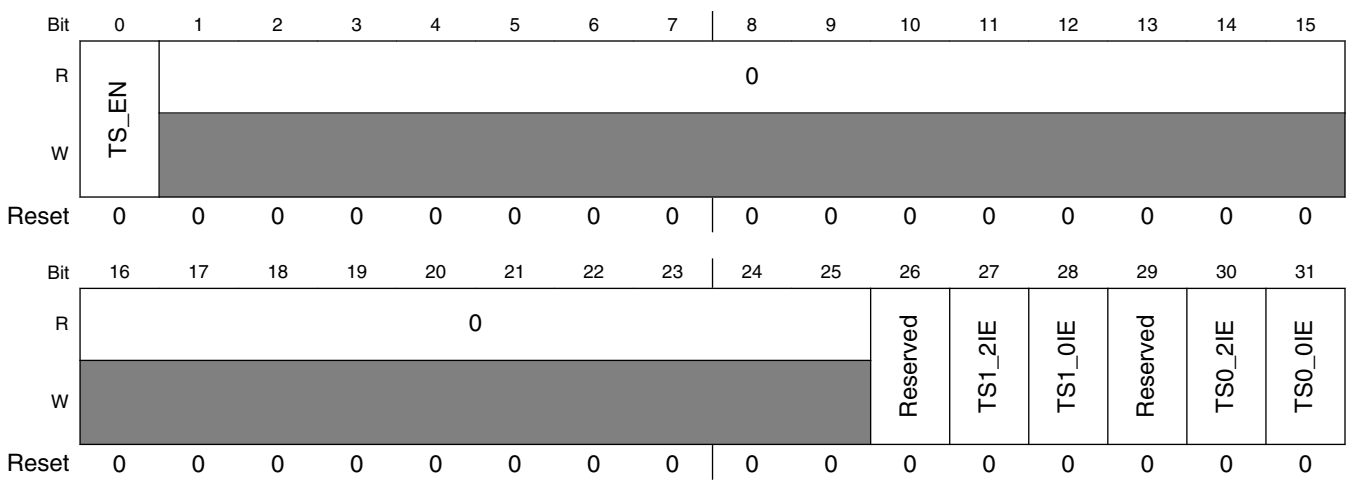
This configuration register contains a set of Interrupt Enable bits that correspond to each of the Temperature Sensor Event Status Registers, ESR_TD (Temperature Event Status register (PMC_ESR_TD)). These bits indicate whether an interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt is to be sent, and a '1' indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

NOTE

This register is reset on a short functional reset. For setting up the interrupt if some temperature event occurs:

1. Enable the MSB of this register, i.e., TS_EN.
2. Assert one of the bits, say TS1_2IE, for interrupt to be sent to CPU if over temp. condition is found for TSENS1. This would help to enable/disable the interrupt functionality of the corresponding temperature sensor flag.

Address: 0h base + 98h offset = 98h



PMC_TS_IER field descriptions

Field	Description
0 TS_EN	Temperature Sensor interrupts Enable 0 Temperature sensor interrupts are not enabled. 1 Enable temperature sensor interrupts.
1–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_TS_IER field descriptions (continued)

Field	Description
26 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
27 TS1_2IE	Temperature Sensor 1 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its high set threshold (flag1), i.e., 150 C. 0 No interrupt. 1 Interrupt occurs.
28 TS1_0IE	Temperature Sensor 1 input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature falls below its low set threshold, i.e., -40 C. 0 No interrupt. 1 Interrupt occurs.
29 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
30 TS0_2IE	Temperature Sensor 0 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its high set threshold (flag1), i.e., 150 C. 0 No interrupt. 1 Interrupt occurs.
31 TS0_0IE	Temperature Sensor 0 input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature falls below its low set threshold, i.e., -40 C. 0 No interrupt. 1 Interrupt occurs.

64.3.9 Temperature Event Status register (PMC_ESR_TD)

This Event Status Register indicates the past state of the two temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared. This register resets on POR.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										Reserved	TEMP1_2	TEMP1_0	Reserved	TEMP0_2	TEMP0_0
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_ESR_TD field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
27 TEMP1_2	Temperature sensor 1 status flag 2. This bit is the temperature status flag associated with the temperature for the temperature sensor 1 point (150C). It is asserted when the temperature exceeds its corresponding threshold. It is cleared when one is written to this bit location.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	NOTE: This bit gets cleared even temperature might not have fallen back. This bit is the indication that temperature has crossed its threshold in the past. If the IER[TS1_2IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_2} is also asserted, a system reset is generated, which clears IER[TS1_2IE] and negate the interrupt request. If REE_TD[TEMP1_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP1_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
28 TEMP1_0	Temperature sensor 1 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor 1 point. It is asserted when the temperature exceeds its corresponding threshold. It is cleared when one is written to this bit location. **NOTE:** This bit gets cleared even temperature might not have fallen back. This bit is the indication that temperature has crossed its threshold in the past. If the IER[TS1_0IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_0] is also asserted, a system reset is generated, which clears IER[TS1_0IE] and negate the interrupt request. If REE_TD[TEMP1_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP1_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
29 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
30 TEMP0_2	Temperature sensor 0 status flag 2. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150C).It is asserted when the temperature exceeds its corresponding threshold. It is cleared when one is written to this bit location. **NOTE:** This bit gets cleared even temperature might not have fallen back. This bit is the indication that temperature has crossed its threshold in the past. If the IER[TS0_2IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_2 is also asserted, a system reset is generated, which clears IER[TS0_2IE] and negate the interrupt request. If REE_TD[TEMP0_2] is asserted and a destructive reset is selected (via RES_TD[TEMP0_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
31 TEMP0_0	Temperature sensor 0 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold. It is cleared when one is written to this bit location. **NOTE:** This bit gets cleared even temperature might not have fallen back. This bit is the indication that temperature has crossed its threshold in the past. If REE_TD[TEMP0_0] is also asserted, a system reset is generated, which clears IER[TS0_0IE] and negate the interrupt request. If REE_TD[TEMP0_0] is asserted and a destructive reset is selected (via RES_TD[TEMP0_0] being 0), then a destructive reset is generated. If RES_TD[TEMP0_0] is set, then a functional reset is generated.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	0 Currently no occurrence. 1 Temperature occurrence detected.

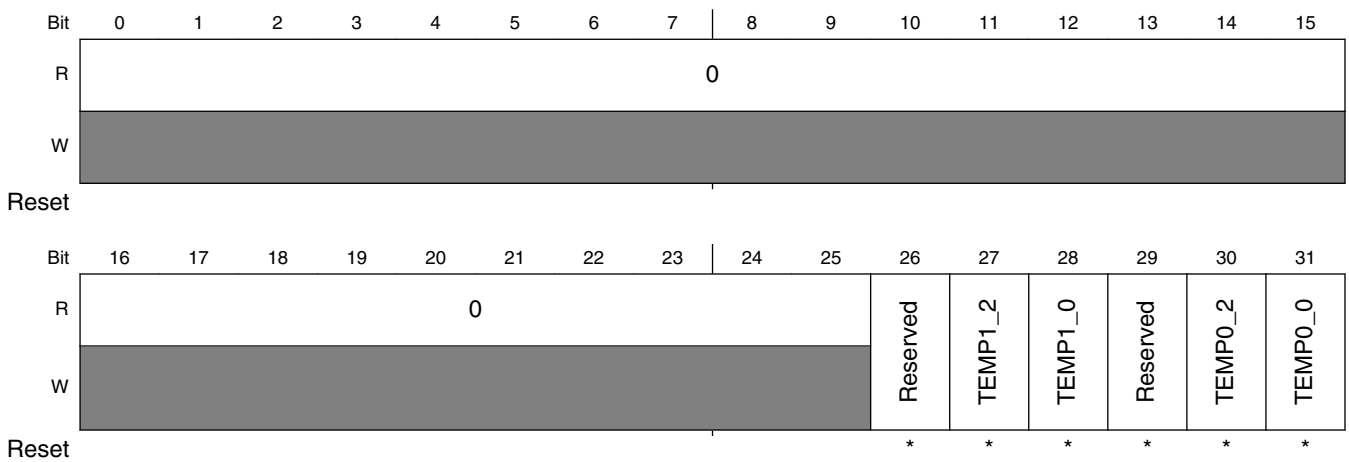
64.3.10 Temperature Reset Event Enable register (PMC_REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

NOTE

Bits in this register are reset on POR.

Address: 0h base + A0h offset = A0h



* Notes:
- TEMP0_0 field: Reset value is configurable by DCF bits.
- TEMP0_2 field: Reset value is configurable by DCF bits.
- Reserved field: Reset value is configurable by DCF bits.
- TEMP1_0 field: Reset value is configurable by DCF bits.
- TEMP1_2 field: Reset value is configurable by DCF bits.
- Reserved field: Reset value is configurable by DCF bits.

PMC_REE_TD field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.

Table continues on the next page...

PMC_REE_TD field descriptions (continued)

Field	Description
27 TEMP1_2	Temperature sensor 1 detect assertion reset enable 2. This bit defines whether high temperature detect assertion (above 150 C) for TSENS1 will generate a system reset. The RES_TD[TEMP1_2] bit determines whether the reset is functional or destructive.
28 TEMP1_0	Temperature sensor 1 detect assertion reset enable 0. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_0] bit determines whether high temperature detect assertion (below -40 C) for TSENS1 will generate a system reset.
29 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
30 TEMP0_2	Temperature sensor 0 detect assertion reset enable 2. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_2] bit determines whether high temperature detect assertion (above 150 C) for TSENS0 will generate a system reset.
31 TEMP0_0	Temperature sensor 0 detect assertion reset enable 0. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_0] bit determines whether high temperature detect assertion (below -40 C) for TSENS0 will generate a system reset.

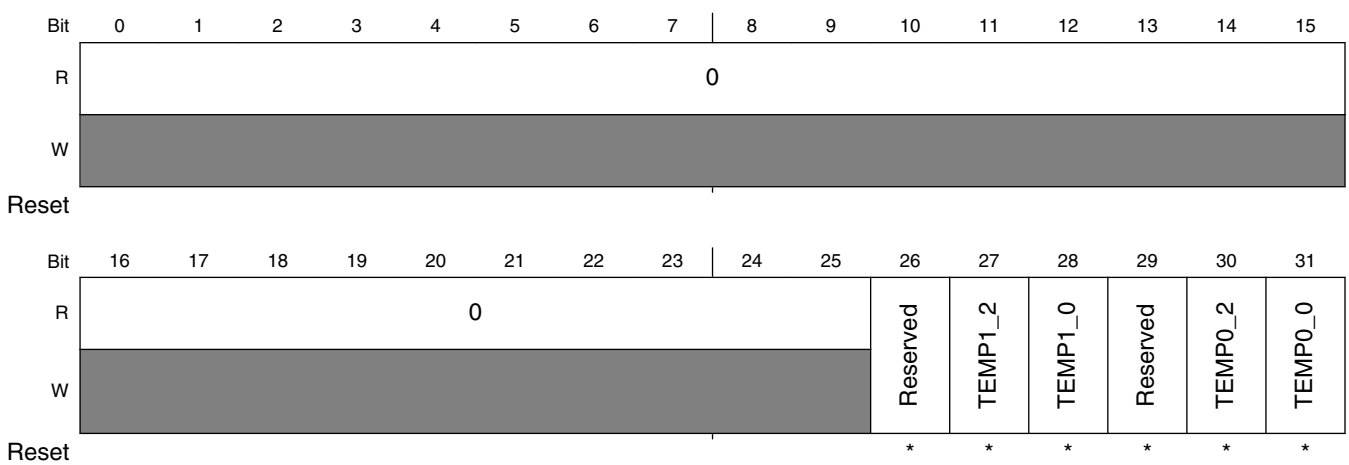
64.3.11 Temperature Reset Event Selection register (PMC_RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is set, then the corresponding temperature sensor event will cause a functional reset to be generated when the selected temperature passes the trigger event else it will cause a destructive reset.

NOTE

Bits in this register are reset on POR.

Address: 0h base + A4h offset = A4h



* Notes:
- TEMP0_0 field: Reset value is configurable by DCF bits.
- TEMP0_2 field: Reset value is configurable by DCF bits.
- Reserved field: Reset value is configurable by DCF bits.
- TEMP1_0 field: Reset value is configurable by DCF bits.
- TEMP1_2 field: Reset value is configurable by DCF bits.
- Reserved field: Reset value is configurable by DCF bits.

PMC_RES_TD field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
27 TEMP1_2	TEMP1_2 Reset Event Select This bit defines whether high temperature detect assertion (above 150 C) for TSENS1 will generate a system reset. The bit determines whether the reset event is enabled.
28 TEMP1_0	TEMP1_0 Reset Event Select This bit defines whether high temperature detect assertion (below -40 C) for TSENS1 will generate a system reset. The bit determines whether the reset event is enabled.
29 Reserved	This field is reserved. **NOTE:** This field must be programmed to 0. If it is programmed to 1, unintended behavior could result.
30 TEMP0_2	TEMP0_2 Reset Event Select This bit defines whether high temperature detect assertion (above 150 C) for TSENS0 will generate a system reset. The bit determines whether the reset event is enabled.
31 TEMP0_0	TEMP0_0 Reset Event Select This bit defineswhether high temperature detect assertion (below -40 C) for TSENS0 will generate a system reset. The bit determines whether the reset event is enabled.

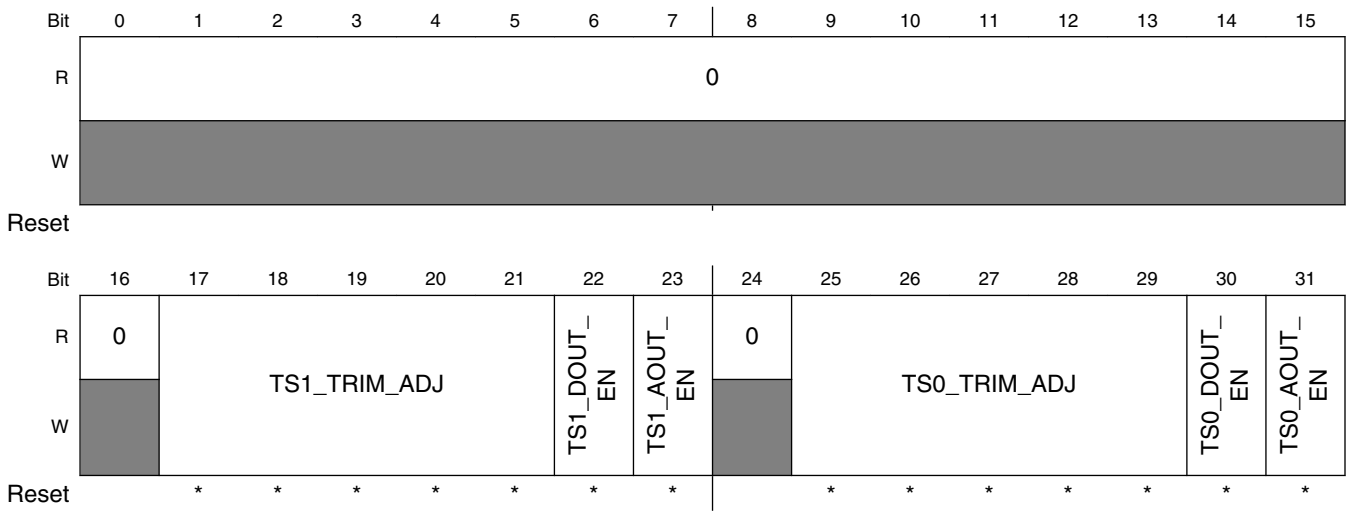
64.3.12 Temperature detector configuration register (PMC_CTL_TD)

This register is reset on functional reset. To disable a temperature sensor, program both TSn_DOUT_EN and TSn_AOUT_EN for that temperature sensor to 0. To enable a temperature sensor, write 1 to both of these fields. Please refer to "Software-controlled Temperature Sensor Trim Adjustment" sub-section in the "Chip-specific Power Management Controller (PMC) information"section.

NOTE

Bits in this register are reset on long functional reset.

Address: 0h base + A8h offset = A8h



* Notes:
- TS0_AOUT_EN field: Reset value is configurable by DCF bits.
- TS0_DOUT_EN field: Reset value is configurable by DCF bits.
- TS0_TRIM_ADJ field: Reset value is configurable by DCF bits.
- TS1_AOUT_EN field: Reset value is configurable by DCF bits.
- TS1_DOUT_EN field: Reset value is configurable by DCF bits.
- TS1_TRIM_ADJ field: Reset value is configurable by DCF bits.

PMC_CTL_TD field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–21 TS1_TRIM_ADJ	Temperature Sensor 1 Trim Adjust value. Customer adjustable trim register. These bits are a signed binary number that is added to the Temp Sensor Trim Value.
22 TS1_DOUT_EN	Temperature Sensor 1 Digital Output Enable.
23 TS1_AOUT_EN	Temperature Sensor 1 Analog Output Enable. Analog Output Enable.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–29 TS0_TRIM_ADJ	Temperature Sensor 0 Trim Adjust value. Customer adjustable trim register. These bits are a signed binary number that is added to the Temp Sensor Trim Value.
30 TS0_DOUT_EN	Temperature Sensor 1 Digital Output Enable.
31 TS0_AOUT_EN	Temperature Sensor 1 Analog Output Enable.

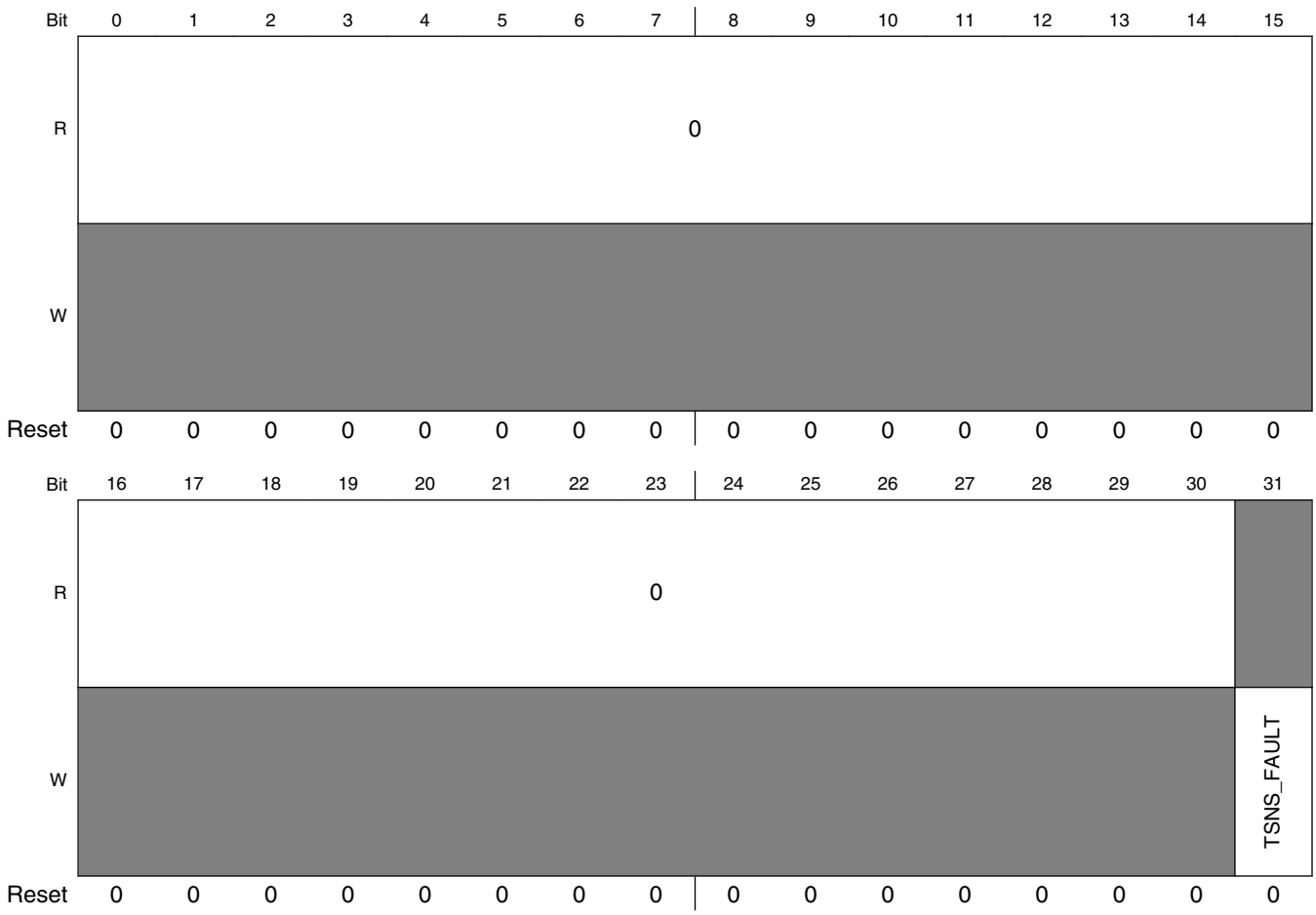
64.3.13 Fault Injection Register (PMC_TS_FIR)

This FCCU Fault Injection register enables the user s/w to artificially generate FCCU faults corresponding to TEMP SENS. A write '1' to these bits generates a single cycle pulsed fault to the FCCU block, this bit automatically gets cleared in the next cycle.

NOTE

This register is reset on short functional reset.

Address: 0h base + B4h offset = B4h



PMC_TS_FIR field descriptions

Field	Description
0–30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 TSNS_FAULT	Temperature Sensor Fault injection. A write to this bit sends a FCCU fault (single cycle) for TEMP SENSOR.

Table continues on the next page...

PMC_TS_FIR field descriptions (continued)

Field	Description
	0 No fault is sent. 1 Single cycle fault injected.

64.4 Functional description

64.4.1 Analog PMC interface

The digital block uses the analog PMC interface to configure trim values for the regulators and LVDs that are in the analog block.

64.4.2 Temperature sensor interface logic

The digital PMC block provides control and configuration information to the two temperature sensors. This information includes digital and analog enables, and trim values. A user adjustable trim control is also provided to fine tune the trimming (for more details, please refer to Temperature Sensor chapter.

64.4.3 POR MC_RGM phase gates

The interface with the RGM block is used during the power on sequence and startup. It provides various phase gates, thereby enabling the MC_RGM to properly complete the reset sequence.

The temperature sensor provides another reset source, based on the under/over temperature faults and the corresponding REE and RES enables.

64.4.4 Flash interface DCF

During power up, trim values for the regulators, LVDs, and HVDs are read from Flash memory by the SSCM, and are then provided to the digital PMC via the DCF interface.

The reset event enable (REE) configurations that are stored in Flash memory are also provided to the PMC via the DCF interface. These REEs configurations are used, by default, to generate various resets to the RGM. Please refer to "Device Configuration Format (DCF) Records" chapter for more details.

64.4.5 FCCU interface

The digital block generates various fault indications to the FCCU block:

- Separate corresponding, active-high, fault indications are generated whenever an LVD or HVD defect is detected in the analog block.
- A fault indication is generated when a self test task results in a failure.
- A fault condition is generated whenever over-temperature or under-temperature events occur in the temperature sensor.

The figures below show the exact logic for the generation fault indications to FCCU block.

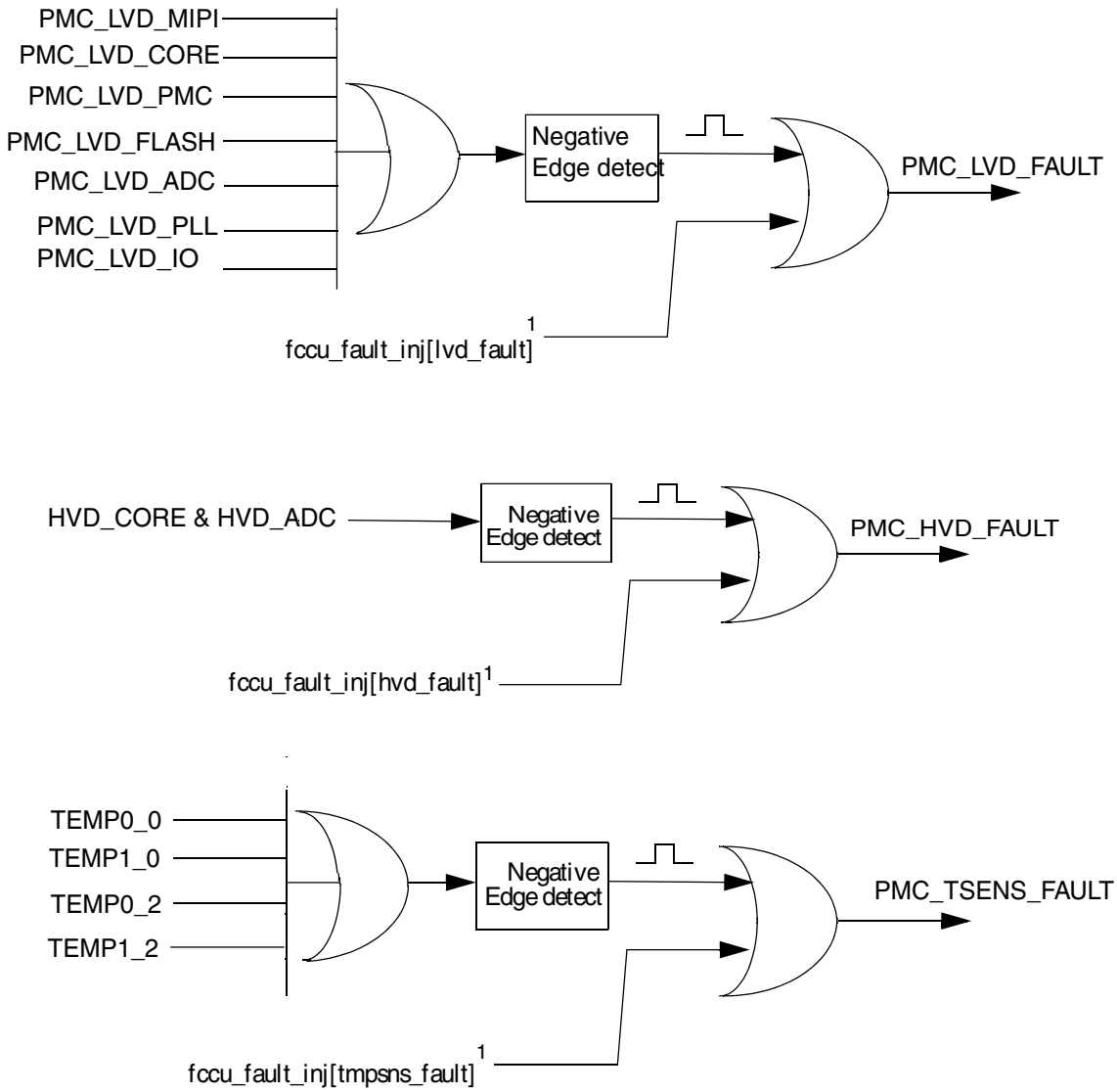


Figure 64-4. Fault generation to FCCU

:

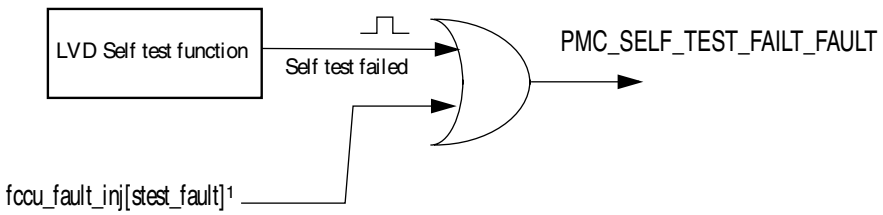


Figure 64-5. FCCU_fault_gen_LVD_Self_Test

[1] See “FCCU Fault Injection Register 0(FIR) for details.

64.4.6 SMPS interface

The digital block has an SMPS Interface:

- The PMC provides an SMPS regulator controlled to be used in conjunction with external components to provide the core supply voltage. The regulator is an asynchronous buck regulator with nominal switching frequency of 1MHz. This frequency can be modulated to improve the EMI generated by the regulator. The start-up behavior of the switching regulator is controlled by the Soft-Start module, that guarantees a controlled increase of the core supply voltage.

64.4.7 LVD and HVD self test mechanism

The PMC implements an LVD/HVD self test mechanism for use in satisfying reliability and robustness goals. The digital block assists and controls the analog block during it's performance of the self test.

Currently, three self test modes (described in the following sections) are supported by the analog block (based on the mode specified by the VD_UTST[ST_MODE] register bit).

The self test status and result are provided in the VD_UTST[ST_DONE, ST_RESULT] register fields, respectively.

64.4.7.1 Default mode

The self test is always performed during start up (i.e power on) or after a destructive reset event. The self test starts during the phase 3 RGM state, when the SSCM has completed reads from the Flash memory.

Meanwhile, the LVD/HVD indication from the analog block is latched in the VD_UTST register. At the end of the self test (i.e. when all of the LVDs/HVDs have been tested), the VD_UTST[ST_DONE] status bit is set to '1' and all of the bits in the VD_UTST register are checked for ‘1’. One or more bits not set to '1' generates a fault indication to the FCCU block, and the VD_UTST[ST_RESULT] register bit is set to ‘0’. In the case where all of the register bits are set to '1', the test is considered to be PASSED, the VD_UTST[ST_RESULT] register bit is set to ‘1’, and no fault indications are provided to the FCCU.

This startup self test function also provides gating of the reset phase3 exit in RGM. This means the RGM will not be able to exit from phase3 until the self test is successfully completed.

64.4.7.2 Software triggered self test

This mode is entered by software by writing '01' to the VD_UTST[ST_MODE] register 2-bit field. The complete self test, as is performed in default mode, is repeated. The difference is that the self test in this mode is initiated by the host while the self test in default mode is started automatically during power on.

NOTE

The LVD self test Time Window Register (STTW) has to be configured appropriately (based on the clock frequency used), before initiating software triggered self test. For example, the configuration (in decimal) for a 66 MHz PMC clock and a 21 us window could be calculated using the following formula:

Freq in MHz x window in us = Decimal value (used to program the counter value in the register).

64.4.7.3 Single voltage detection test

This mode is entered by software writing '10' to the VD_ST_CTRL[ST_MODE] register 2-bit field. The test completes after the prescribed time gap has elapsed. The PASS/FAIL result is then updated based on the corresponding LVD output tripping, or not.

NOTE

The LVD Self Test Time Window Register (STTW) must to be configured appropriately as in Software triggered self test mode.

NOTE

During the running of the LVD self test, the LVD or HVD output, which corresponds to the LVD or HVD monitor on which self test is running, is masked. Therefore, they will have no effect on POR generation, or in any reset and/or fault generation, when self test is running on that particular monitor. Also, it is possible to mask all the LVDs and HVDs during self test (configurable through DCF sheet). Please refer to device-specific RM on this.

Chapter 65
Power Control Unit (MC_PCU)

65.1 Introduction

65.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

The following figure depicts the MC_PCU block diagram.

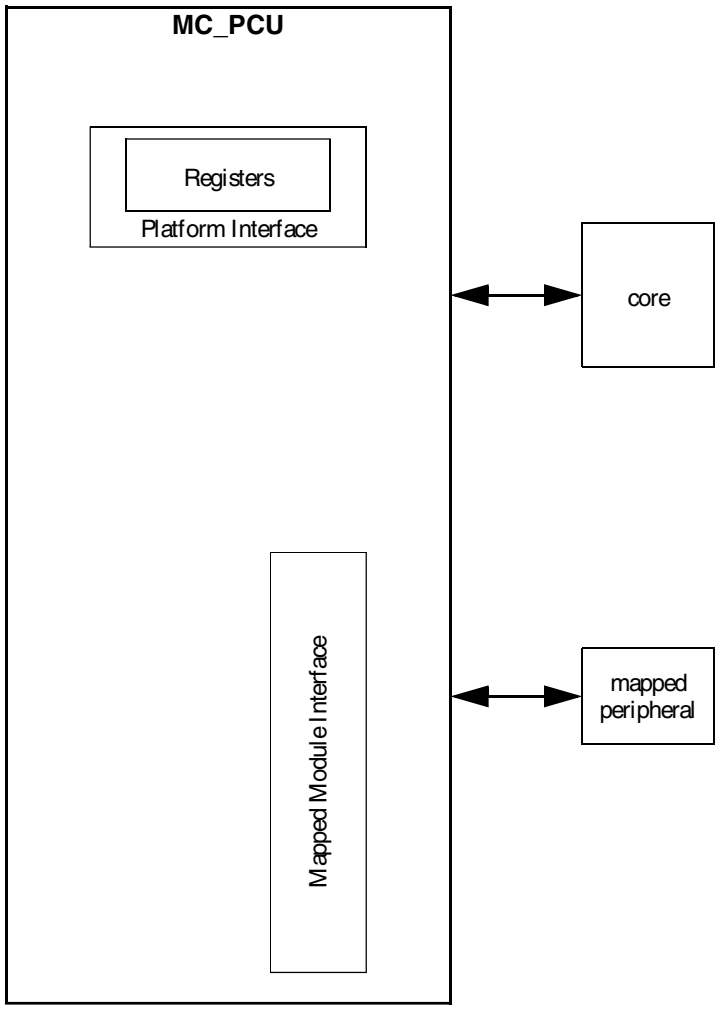


Figure 65-1. MC_PCU block diagram

65.1.2 Features

The MC_PCU maps the PMC registers to the MC_PCU address space.

65.2 External signal description

The MC_PCU has no connections to any external pins.

65.3 Memory Map and Registers

Any access to unused registers as well as write accesses to read-only registers:

- Does not change register content
- Cause a transfer error

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian.

MC_PCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
40	Power Domain Status Register (MC_PCU_PSTAT)	32	R	0000_0001h	65.3.1/3371

65.3.1 Power Domain Status Register (MC_PCU_PSTAT)

This register reflects the power status of all available power domains. It can be read in user and supervisor mode.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_PCU_PSTAT field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

Chapter 66
Mode Entry Module (MC_ME)

66.1 Introduction

66.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

The following figure shows the MC_ME block diagram.

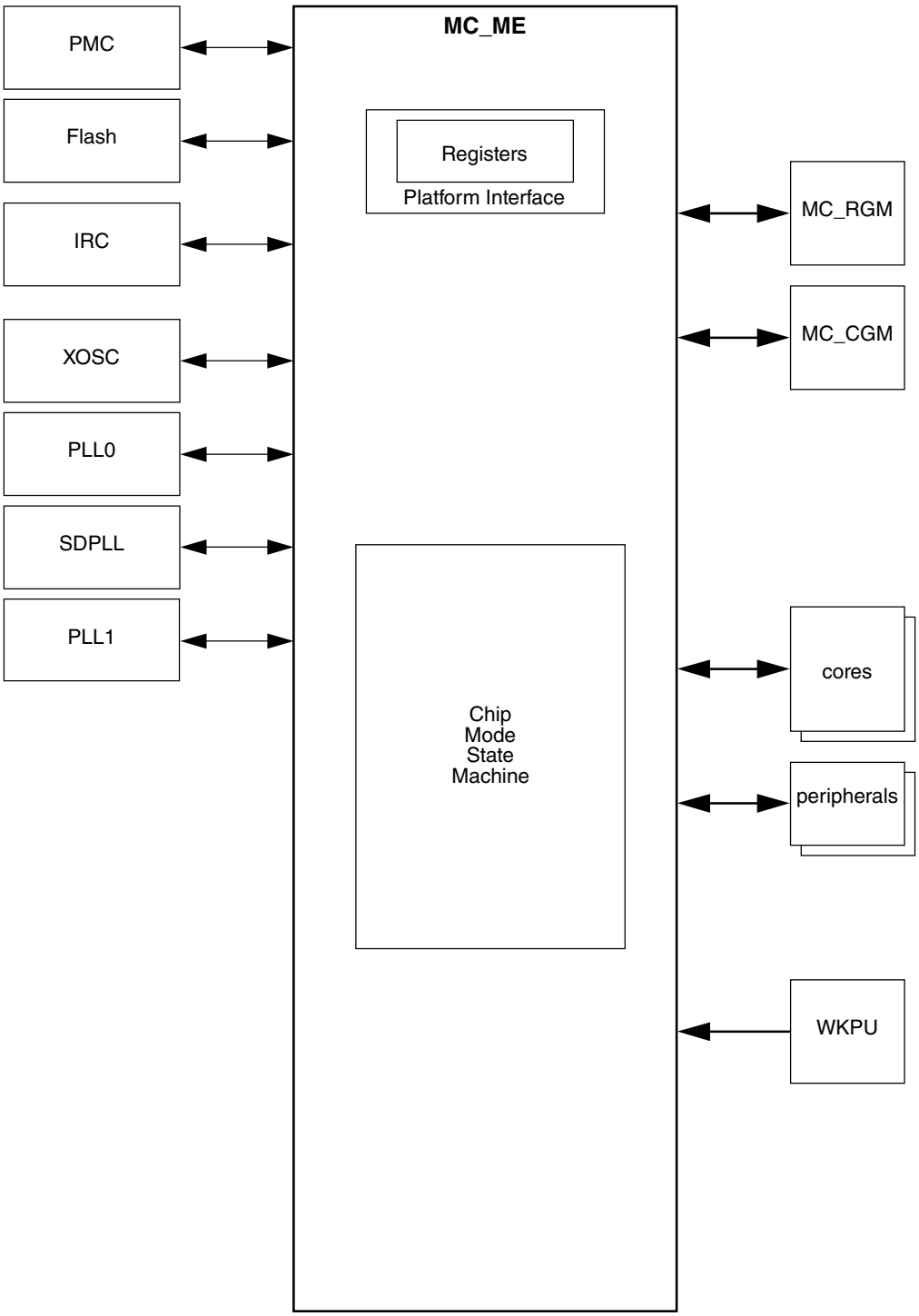


Figure 66-1. MC_ME Block Diagram

66.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register for:
  - switching between software-running modes
  - switching from software-running modes to low-power modes
  - switching to a mode to run diagnostics after a failure has been detected
  - triggering a device ‘functional’ reset which does not cause a start-up self test to run
  - triggering a device ‘destructive’ reset which may cause a start-up self test to run
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers
- additional core clock gating and boot address control based on the ME_CCTLn and ME_CADDRn registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

66.1.3 Modes of Operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0…3, HALT0, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0…3 are the chip software running modes.

The following table describes the MC_ME modes.

Table 66-1. MC_ME Mode Descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0…3	system reset assertion, RUN0…3, TEST via software, SAFE via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0…3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, SAFE via software or hardware failure, DRUN via software
RUN0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0…3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0…3 modes, HALT0, STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled[1]. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled[1]. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event or wakeup event

1. Along with the Cores, clocks to the below peripherals are also disabled in HALT0 and STOP0 modes.
   - Core_0/1/2 related TCM and DMEM logic
   - SWT_0/1/2
   - STM_0/1/2
   - SMPU_0/1
   - EIM
   - PCM
   - AXBS_0/1
   - XBIC_0/1
   - AIPS_0/1
   - CMU_11
   - SEMA42
   - CSE
   - TDM
   - PFLASH/c55fmc
   - PRAMC
   - DMA and related interface to AXBS

- SPT-AHB/SIPI-AHB/FEC-AHB/Flexray-AHB interface
- NAL/NPC/NXMC_0/1/2

66.2 External Signal Description

The MC_ME has no connections to any external pins.

66.3 Memory Map and Registers

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

NOTE

All registers are read only in user mode

MC_ME memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Status Register (MC_ME_GS)	32	R	3013_0010h	66.3.1/3381
4	Mode Control Register (MC_ME_MCTL)	32	R/W	3000_A50Fh	66.3.2/3384
8	Mode Enable Register (MC_ME_ME)	32	R/W	0000_801Dh	66.3.3/3386

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C	Interrupt Status Register (MC_ME_IS)	32	w1c	0000_0000h	66.3.4/3388
10	Interrupt Mask Register (MC_ME_IM)	32	R/W	0000_0000h	66.3.5/3389
14	Invalid Mode Transition Status Register (MC_ME_IMTS)	32	w1c	0000_0000h	66.3.6/3391
18	Debug Mode Transition Status Register (MC_ME_DMTS)	32	R	0000_0000h	66.3.7/3392
20	RESET Mode Configuration Register (MC_ME_RESET_MC)	32	R	0013_0010h	66.3.8/3396
24	TEST Mode Configuration Register (MC_ME_TEST_MC)	32	R/W	0013_0010h	66.3.9/3398
28	SAFE Mode Configuration Register (MC_ME_SAFE_MC)	32	R	0093_0010h	66.3.10/ 3401
2C	DRUN Mode Configuration Register (MC_ME_DRUN_MC)	32	R/W	0013_0010h	66.3.11/ 3403
30	RUN0 3 Mode Configuration Register (MC_ME_RUN0_MC)	32	R/W	0013_0010h	66.3.12/ 3405
34	RUN0 3 Mode Configuration Register (MC_ME_RUN1_MC)	32	R/W	0013_0010h	66.3.12/ 3405
38	RUN0 3 Mode Configuration Register (MC_ME_RUN2_MC)	32	R/W	0013_0010h	66.3.12/ 3405
3C	RUN0 3 Mode Configuration Register (MC_ME_RUN3_MC)	32	R/W	0013_0010h	66.3.12/ 3405
40	HALT0 Mode Configuration Register (MC_ME_HALT0_MC)	32	R/W	0013_0010h	66.3.13/ 3408
48	STOP0 Mode Configuration Register (MC_ME_STOP0_MC)	32	R/W	0013_0010h	66.3.14/ 3410
60	Peripheral Status Register 0 (MC_ME_PS0)	32	R	0000_0000h	66.3.15/ 3413
64	Peripheral Status Register 1 (MC_ME_PS1)	32	R	0000_0000h	66.3.16/ 3415
68	Peripheral Status Register 2 (MC_ME_PS2)	32	R	0000_0000h	66.3.17/ 3417
6C	Peripheral Status Register 3 (MC_ME_PS3)	32	R	0000_0000h	66.3.18/ 3419
70	Peripheral Status Register 4 (MC_ME_PS4)	32	R	0000_0000h	66.3.19/ 3422
74	Peripheral Status Register 5 (MC_ME_PS5)	32	R	0000_0000h	66.3.20/ 3424
78	Peripheral Status Register 6 (MC_ME_PS6)	32	R	0000_0000h	66.3.21/ 3426
7C	Peripheral Status Register 7 (MC_ME_PS7)	32	R	0000_0000h	66.3.22/ 3427
80	Run Peripheral Configuration Register (MC_ME_RUN_PC0)	32	R/W	0000_0000h	66.3.23/ 3429
84	Run Peripheral Configuration Register (MC_ME_RUN_PC1)	32	R/W	0000_0000h	66.3.23/ 3429
88	Run Peripheral Configuration Register (MC_ME_RUN_PC2)	32	R/W	0000_0000h	66.3.23/ 3429

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C	Run Peripheral Configuration Register (MC_ME_RUN_PC3)	32	R/W	0000_0000h	66.3.23/ 3429
90	Run Peripheral Configuration Register (MC_ME_RUN_PC4)	32	R/W	0000_0000h	66.3.23/ 3429
94	Run Peripheral Configuration Register (MC_ME_RUN_PC5)	32	R/W	0000_0000h	66.3.23/ 3429
98	Run Peripheral Configuration Register (MC_ME_RUN_PC6)	32	R/W	0000_0000h	66.3.23/ 3429
9C	Run Peripheral Configuration Register (MC_ME_RUN_PC7)	32	R/W	0000_0000h	66.3.23/ 3429
A0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC0)	32	R/W	0000_0000h	66.3.24/ 3430
A4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC1)	32	R/W	0000_0000h	66.3.24/ 3430
A8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC2)	32	R/W	0000_0000h	66.3.24/ 3430
AC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC3)	32	R/W	0000_0000h	66.3.24/ 3430
B0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC4)	32	R/W	0000_0000h	66.3.24/ 3430
B4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC5)	32	R/W	0000_0000h	66.3.24/ 3430
B8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC6)	32	R/W	0000_0000h	66.3.24/ 3430
BC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC7)	32	R/W	0000_0000h	66.3.24/ 3430
C9	LFAST_0 Peripheral Control Register (MC_ME_PCTL9)	8	R/W	00h	66.3.25/ 3431
CB	SIPI_0 Peripheral Control Register (MC_ME_PCTL11)	8	R/W	00h	66.3.26/ 3432
CC	Ethernet_0 Peripheral Control Register (MC_ME_PCTL12)	8	R/W	00h	66.3.27/ 3433
DE	PIT_RTC_0 Peripheral Control Register (MC_ME_PCTL30)	8	R/W	00h	66.3.28/ 3434
DF	PIT_RTC_1 Peripheral Control Register (MC_ME_PCTL31)	8	R/W	00h	66.3.29/ 3435
E4	DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)	8	R/W	00h	66.3.30/ 3436
E6	CRC_0 Peripheral Control Register (MC_ME_PCTL38)	8	R/W	00h	66.3.31/ 3437
ED	JTAGM Peripheral Control Register (MC_ME_PCTL45)	8	R/W	00h	66.3.32/ 3438
F1	DTS Peripheral Control Register (MC_ME_PCTL49)	8	R/W	00h	66.3.33/ 3439

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FD	MIPI Peripheral Control Register (MC_ME_PCTL61)	8	R/W	00h	66.3.34/ 3440
10D	FLEXCAN_2 Peripheral Control Register (MC_ME_PCTL77)	8	R/W	00h	66.3.35/ 3441
10E	FLEXCAN_1 Peripheral Control Register (MC_ME_PCTL78)	8	R/W	00h	66.3.36/ 3442
10F	FLEXCAN_0 Peripheral Control Register (MC_ME_PCTL79)	8	R/W	00h	66.3.37/ 3443
11B	LINFlex_1 Peripheral Control Register (MC_ME_PCTL91)	8	R/W	00h	66.3.38/ 3444
122	DSPI_1 Peripheral Control Register (MC_ME_PCTL98)	8	R/W	00h	66.3.39/ 3445
124	IIC_2 Peripheral Control Register (MC_ME_PCTL100)	8	R/W	00h	66.3.40/ 3446
126	IIC_1 Peripheral Control Register (MC_ME_PCTL102)	8	R/W	00h	66.3.41/ 3447
12B	Flexray Peripheral Control Register (MC_ME_PCTL107)	8	R/W	00h	66.3.42/ 3448
13E	SAR_ADC_1 Peripheral Control Register (MC_ME_PCTL126)	8	R/W	00h	66.3.43/ 3449
149	ETIMER_1 Peripheral Control Register (MC_ME_PCTL137)	8	R/W	00h	66.3.44/ 3450
152	DMAMUX_1 Peripheral Control Register (MC_ME_PCTL146)	8	R/W	00h	66.3.45/ 3451
154	CRC_1 Peripheral Control Register (MC_ME_PCTL148)	8	R/W	00h	66.3.46/ 3453
17B	SPT Peripheral Control Register (MC_ME_PCTL187)	8	R/W	00h	66.3.47/ 3454
17C	RADAR_CTE Peripheral Control Register (MC_ME_PCTL188)	8	R/W	00h	66.3.48/ 3455
191	DSPI_2 Peripheral Control Register (MC_ME_PCTL209)	8	R/W	00h	66.3.49/ 3456
1AD	SAR_ADC_0 Peripheral Control Register (MC_ME_PCTL237)	8	R/W	00h	66.3.50/ 3457
1AE	WGM_0 Peripheral Control Register (MC_ME_PCTL238)	8	R/W	00h	66.3.51/ 3458
1B5	ETIMER_2 Peripheral Control Register (MC_ME_PCTL245)	8	R/W	00h	66.3.52/ 3459
1BB	CTU_0 Peripheral Control Register (MC_ME_PCTL251)	8	R/W	00h	66.3.53/ 3460
1BF	FlexPWM_0 Peripheral Control Register (MC_ME_PCTL255)	8	R/W	00h	66.3.54/ 3461
1C0	Core Status Register (MC_ME_CS)	32	R	See section	66.3.55/ 3462

Table continues on the next page...

MC_ME memory map (continued)

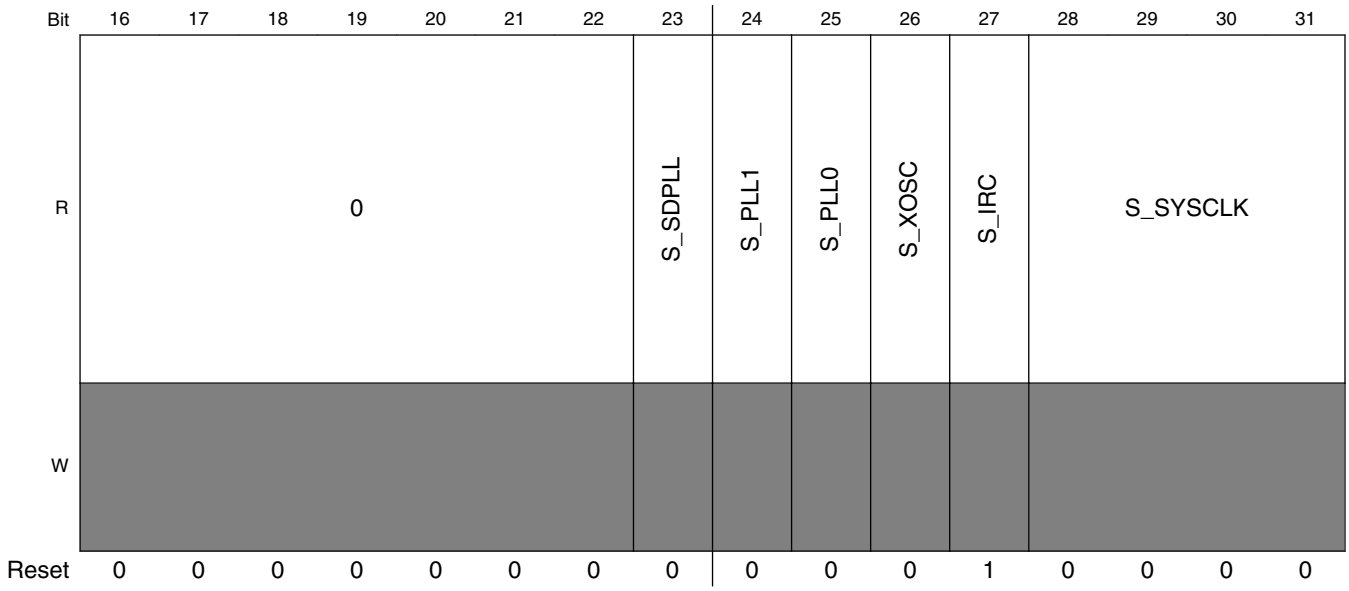
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C6	Core Control Register (MC_ME_CCTL1)	16	R/W	See section	66.3.56/ 3463
1C8	Core Control Register (MC_ME_CCTL2)	16	R/W	See section	66.3.57/ 3465
1CA	Core Control Register (MC_ME_CCTL3)	16	R/W	0000h	66.3.58/ 3466
1E4	Core Control Registers (MC_ME_CADDR1)	32	R/W	See section	66.3.59/ 3468
1E8	Core Control Registers (MC_ME_CADDR2)	32	R/W	See section	66.3.60/ 3469
1EC	Core Control Registers (MC_ME_CADDR3)	32	R/W	See section	66.3.61/ 3469

66.3.1 Global Status Register (MC_ME_GS)

This register contains global mode status.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	0	0		S_PDO	0		S_MVR	0		S_FLA	
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_GS field descriptions

Field	Description
0–3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_PDO	Output power-down status - This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off.

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 S_FLA	Flash availability status 00 Reserved 01 Reserved 10 Reserved 11 Flash is in normal mode and available for use
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SDPLL	SDPLL status 0 SDPLL is not stable 1 SDPLL is providing a stable clock
24 S_PLL1	PLL1 status 0 PLL1 is not stable 1 PLL1 is providing a stable clock
25 S_PLL0	PLL0 status 0 PLL0 is not stable 1 PLL0 is providing a stable clock
26 S_XOSC	XOSC status **NOTE:** S_XOSC alone should not be relied on. To get the current status of XOSC also check CMU0_ISR[OLRI] flag. XOSC is stable when MC_ME_GS[S_XOSC] is 1 and OLRI flag = 0 0 XOSC is not stable 1 XOSC is providing a stable clock
27 S_IRC	IRC status 0 IRC is not stable 1 IRC is providing a stable clock
28–31 S_SYSCLK	System clock switch status - These bits specify the system clock currently used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

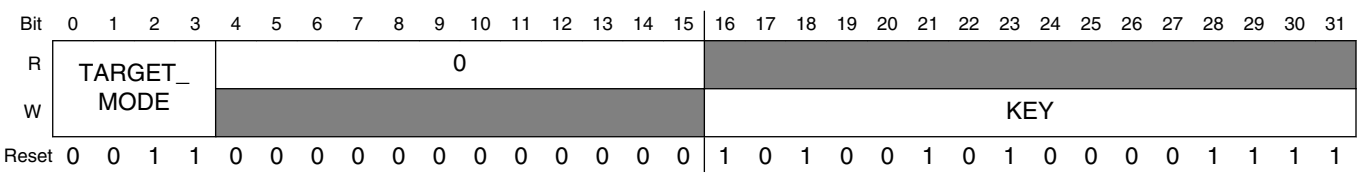
66.3.2 Mode Control Register (MC_ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Address: 0h base + 4h offset = 4h



MC_ME_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target chip mode - These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0…3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0

Table continues on the next page...

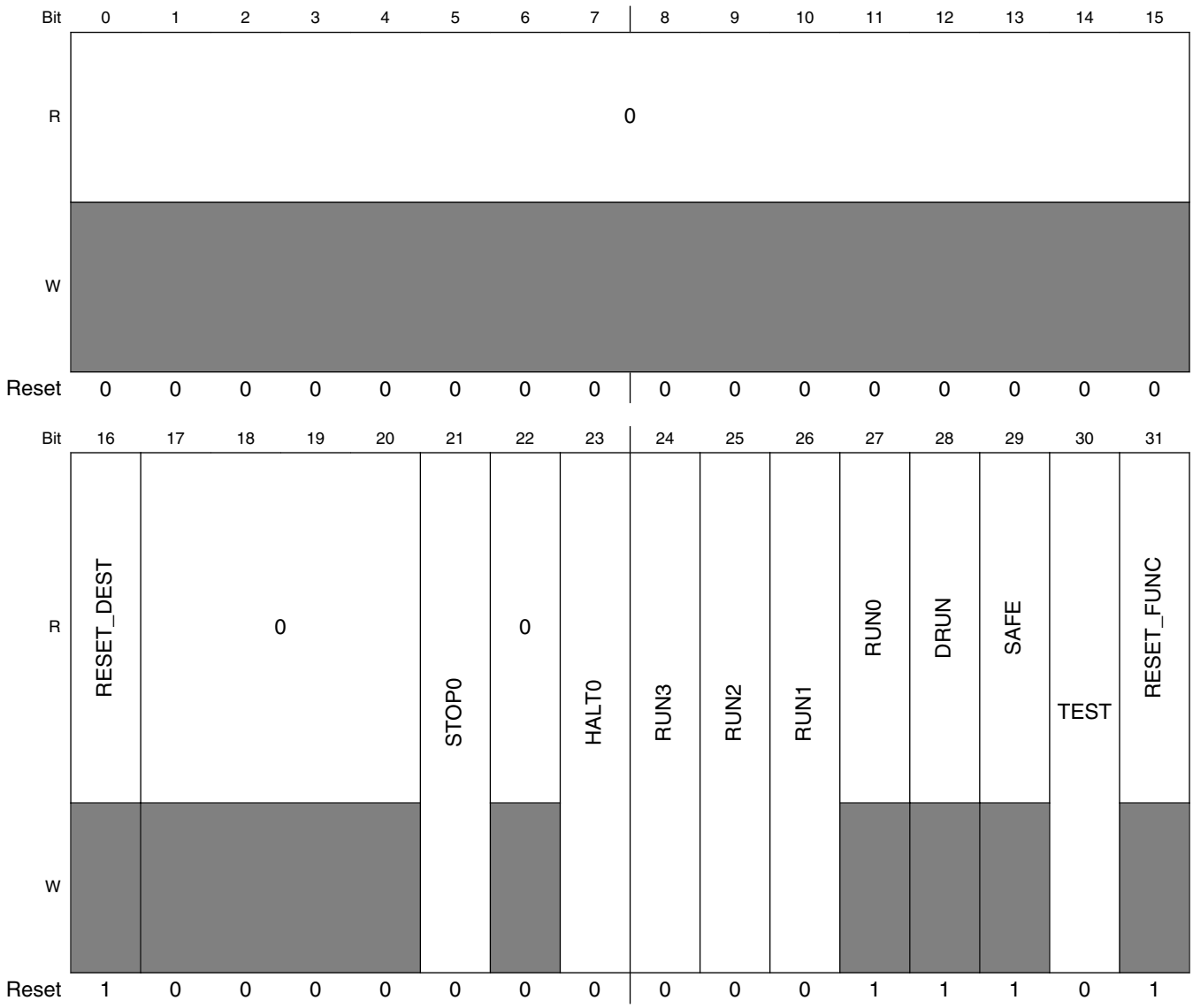
MC_ME_MCTL field descriptions (continued)

Field	Description
	0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 RESET (triggers a 'destructive' reset event)
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	Control key - These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

66.3.3 Mode Enable Register (MC_ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET_FUNC, SAFE, DRUN, RUN0, and RESET_DEST modes are always enabled.

Address: 0h base + 8h offset = 8h



MC_ME_ME field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

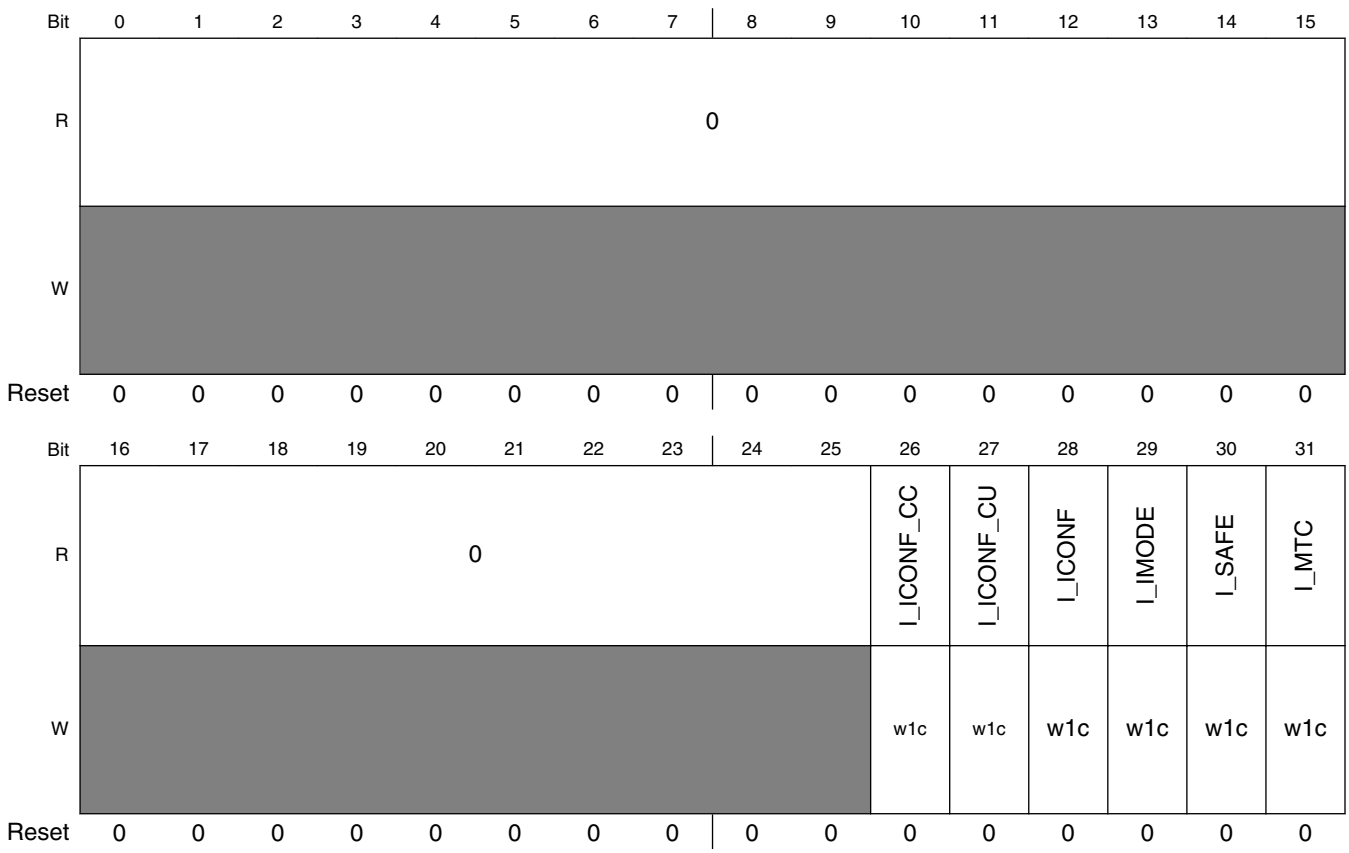
MC_ME_ME field descriptions (continued)

Field	Description
16 RESET_DEST	'destructive' RESET mode enable 1 'destructive' RESET mode is enabled
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	'functional' RESET mode enable 1 'functional' RESET mode is enabled

66.3.4 Interrupt Status Register (MC_ME_IS)

This register provides the current interrupt status.

Address: 0h base + Ch offset = Ch



MC_ME_IS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) - This bit is set if a write access to one of the ME_CADDRn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
27 I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) - This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending

Table continues on the next page...

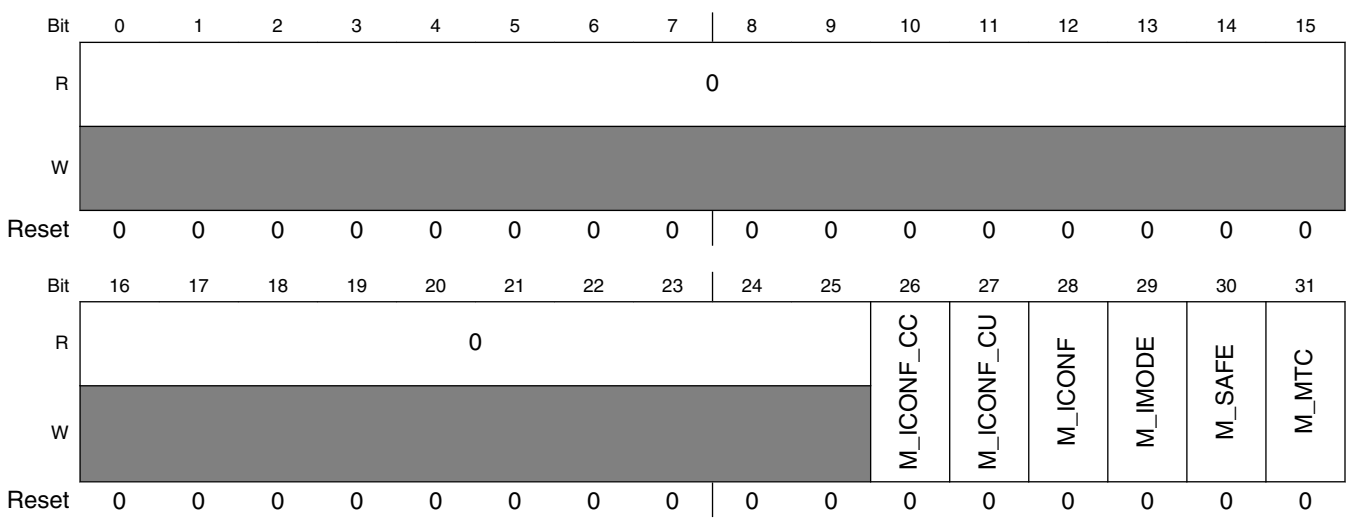
MC_ME_IS field descriptions (continued)

Field	Description
28 I_ICONF	Invalid mode configuration interrupt - This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt - This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt - This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt - This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

66.3.5 Interrupt Mask Register (MC_ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0h base + 10h offset = 10h



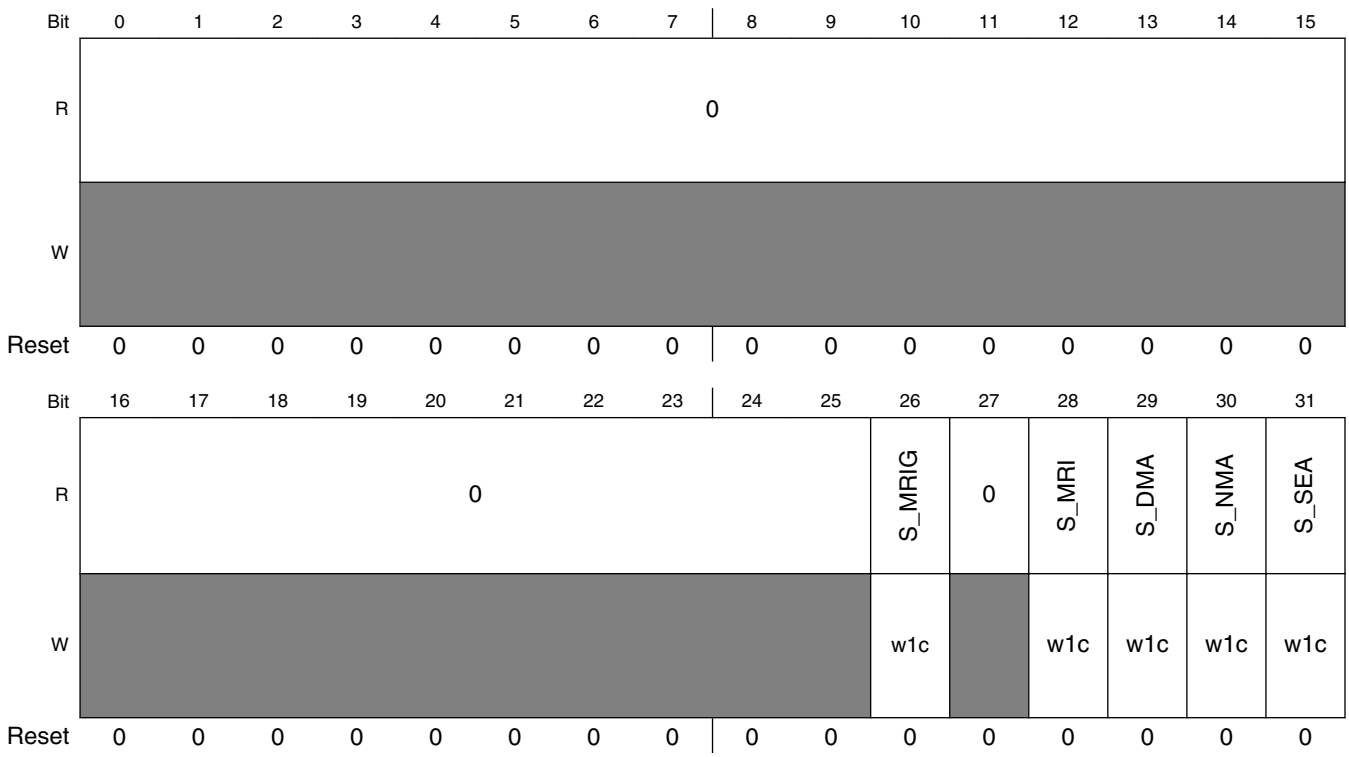
MC_ME_IM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

66.3.6 Invalid Mode Transition Status Register (MC_ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0h base + 14h offset = 14h



MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	S_MRIG Mode Request Ignored status — This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. It is cleared by writing a ‘1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal This bit is also set if a transition to STOP/HALT mode is requested when a wakeup is active.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_MRI	Mode Request Illegal status - This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode

Table continues on the next page...

MC_ME_IMTS field descriptions (continued)

Field	Description
29 S_DMA	Disabled Mode Access status - This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status - This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status - This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

66.3.7 Debug Mode Transition Status Register (MC_ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

NOTE

The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0				MPH_BUSY	0		PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	0	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_DMTS field descriptions

Field	Description
0–3 PREVIOUS_ MODE	Previous chip mode - These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator - This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its handshake. 0 Handshake is not busy 1 Handshake is busy
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PMC_PROG	MC_PCU Mode Change in Progress indicator - This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator - This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress - This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress - This bit is set while the progressive system clock switching process is in progress. 0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator - This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	0 A SAFE mode request is not active 1 A SAFE mode request is active
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
19 IRC_SC	IRC State Change during mode transition indicator - This bit is set when the IRC is requested to change its power up/down state. It is cleared when the IRC has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator - This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A 'secondary clock source' is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status - 0 No system clock source switching is pending 1 A system clock source switching is pending
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224...255[1] - This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
25 CDP_PRPH_192_223	Clock Disable Process Pending status for Peripherals 192...223 - This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
26 CDP_PRPH_ 160_191	Clock Disable Process Pending status for Peripherals 160...191 - This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
27 CDP_PRPH_ 128_159	Clock Disable Process Pending status for Peripherals 128...159 - This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_ 127	Clock Disable Process Pending status for Peripherals 96...127 - This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_ 95	Clock Disable Process Pending status for Peripherals 64...95 - This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
30 CDP_PRPH_32_ 63	Clock Disable Process Pending status for Peripherals 32...63 - This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_ 31	Clock Disable Process Pending status for Peripherals 0...31 - This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. Peripheral n corresponds to the ME_PCTLn register. Please refer to the module memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

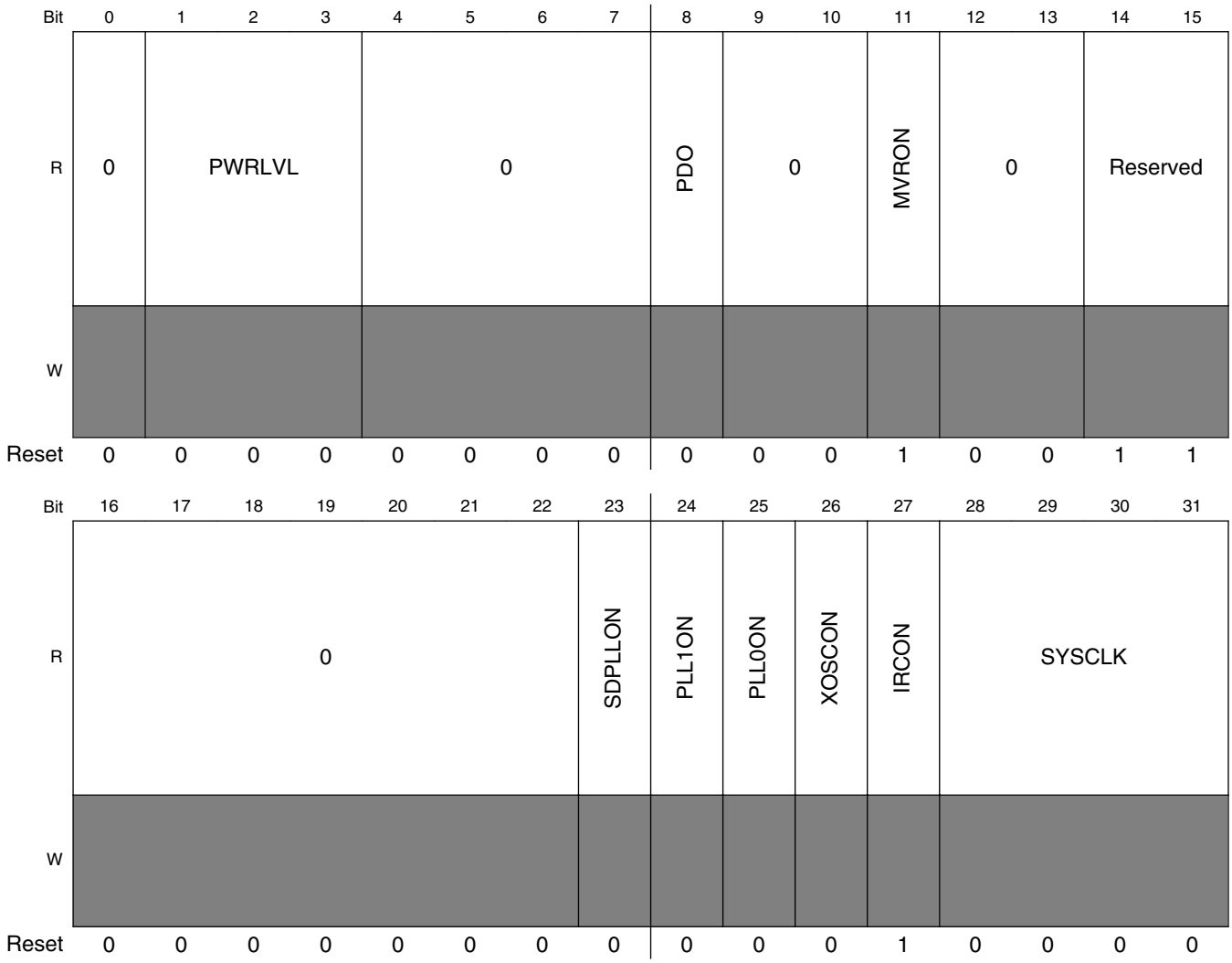
66.3.8 RESET Mode Configuration Register (MC_ME_RESET_MC)

This register configures system behavior during RESET mode.

NOTE

The following configuration values are set according to the chip configuration: XOSCON

Address: 0h base + 20h offset = 20h



MC_ME_RESET_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control-This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. — 11 Flash is in normal mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off
24 PLL1ON	PLL1 control 0 PLL1 is switched off
25 PLL0ON	PLL0 control 0 PLL0 is switched off
26 XOSCON	XOSC control 0 XOSC is switched off
27 IRCON	IRC control 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

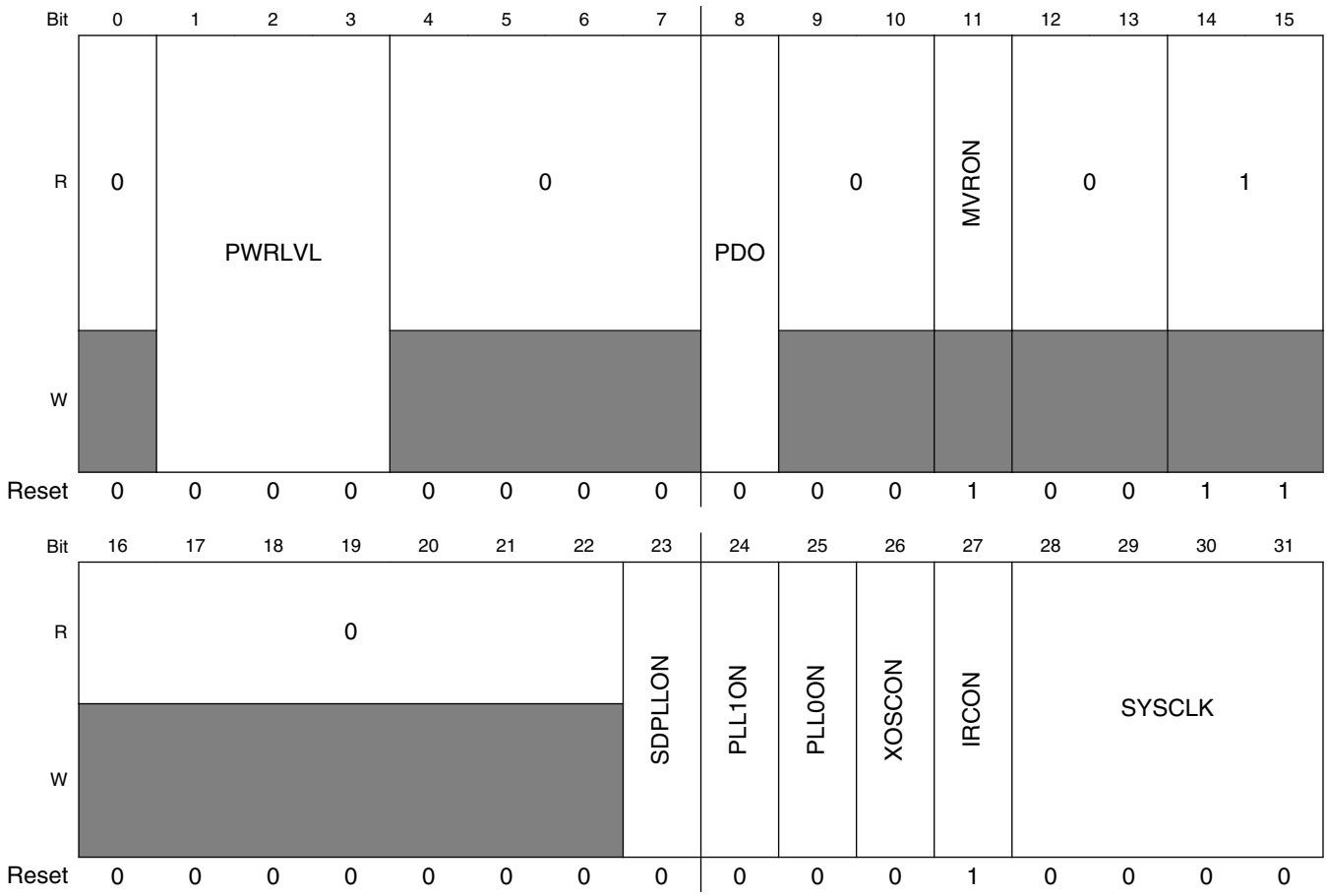
66.3.9 TEST Mode Configuration Register (MC_ME_TEST_MC)

This register configures system behavior during TEST mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 24h offset = 24h



MC_ME_TEST_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_TEST_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1. — 11 Flash is in normal mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off 1 SDPLL is switched on
24 PLL1ON	PLL1 control 0 PLL1 is switched off 1 PLL1is switched on
25 PLL0ON	PLL0 control 0 PLL0 is switched off 1 PLL0 is switched on
26 XOSCON	XOSC control 0 XOSC is switched off 1 XOSC is switched on
27 IRCON	IRC control 0 IRC is switched off 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_TEST_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

66.3.10 SAFE Mode Configuration Register (MC_ME_SAFE_MC)

This register configures system behavior during SAFE mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		Reserved	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							SDPLLON	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_SAFE_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off
24 PLL1ON	PLL1 control 0 PLL1 is switched off
25 PLL0ON	PLL0 control 0 PLL0 is switched off
26 XOSCON	XOSC control 0 XOSC is switched off
27 IRCON	IRC control 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved

Table continues on the next page...

MC_ME_SAFE_MC field descriptions (continued)

Field	Description
	1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

66.3.11 DRUN Mode Configuration Register (MC_ME_DRUN_MC)

This register configures system behavior during DRUN mode.

NOTE

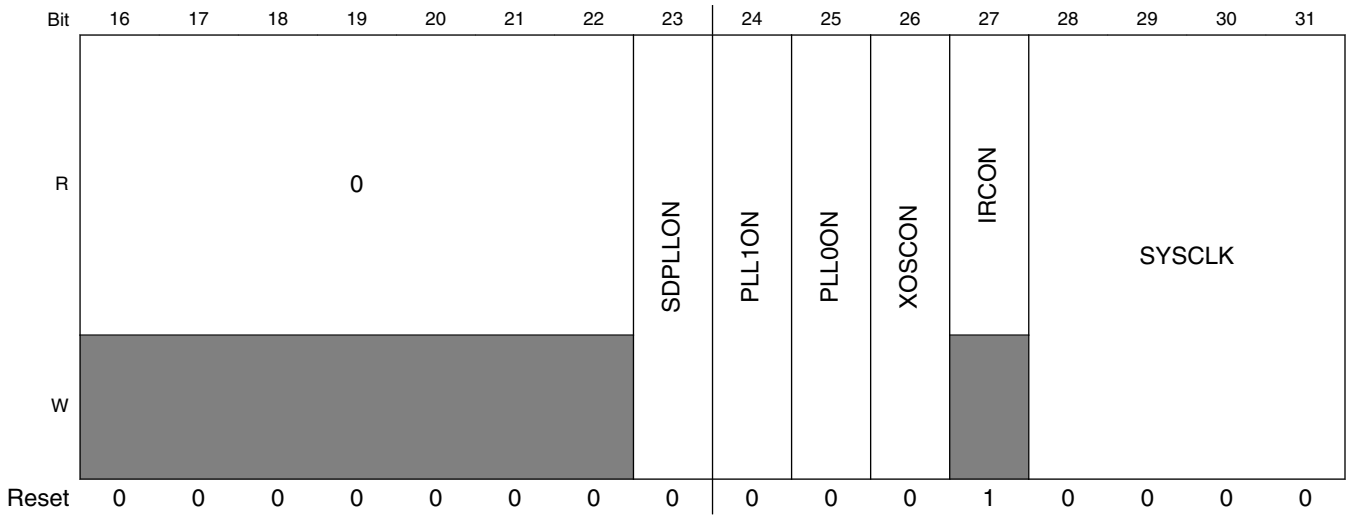
Byte write accesses are not allowed to this register.

NOTE

The following configuration values are set according to the chip configuration: XOSCON

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_DRUN_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off 1 SDPLL is switched on
24 PLL1ON	PLL1 control

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	0 PLL1 is switched off 1 PLL1 is switched on
25 PLL0ON	PLL0 control 0 PLL0 is switched off 1 PLL0 is switched on
26 XOSCON	XOSC control 0 XOSC is switched off 1 XOSC is switched on
27 IRCON	IRC control 0 IRC is switched off 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

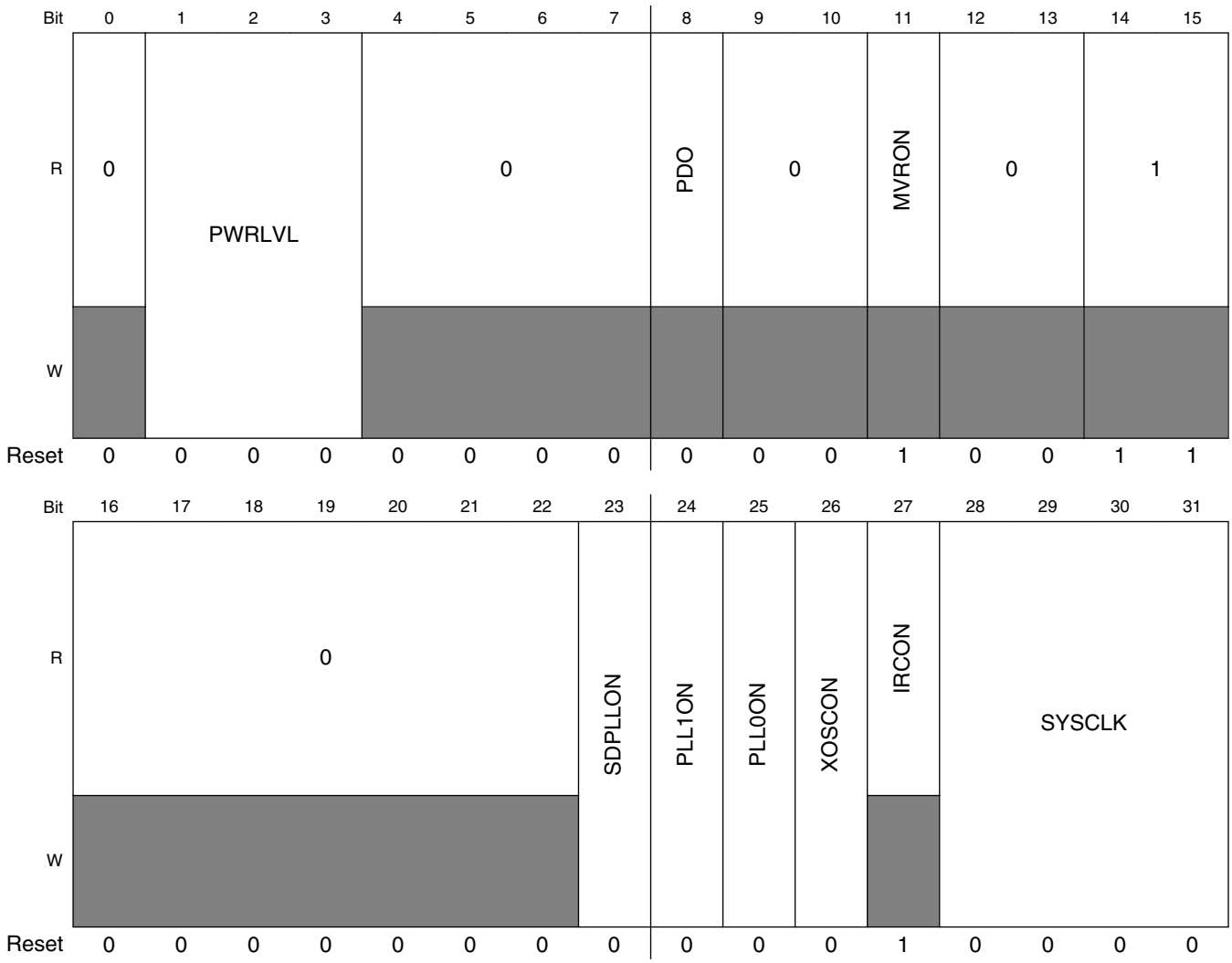
66.3.12 RUN0 3 Mode Configuration Register (MC_ME_RUN*n*_MC)

This register configures system behavior during RUN0…3 modes.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 30h offset + (4d × i), where i=0d to 3d



MC_ME_RUN*n*_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN*n*_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off 1 SDPLL is switched on
24 PLL1ON	PLL1 control 0 PLL1 is switched off 1 PLL1 is switched on
25 PLL0ON	PLL0 control 0 PLL0 is switched off 1 PLL0 is switched on
26 XOSCON	XOSC control 0 XOSC is switched off 1 XOSC is switched on
27 IRCON	IRC control 0 IRC is switched off 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

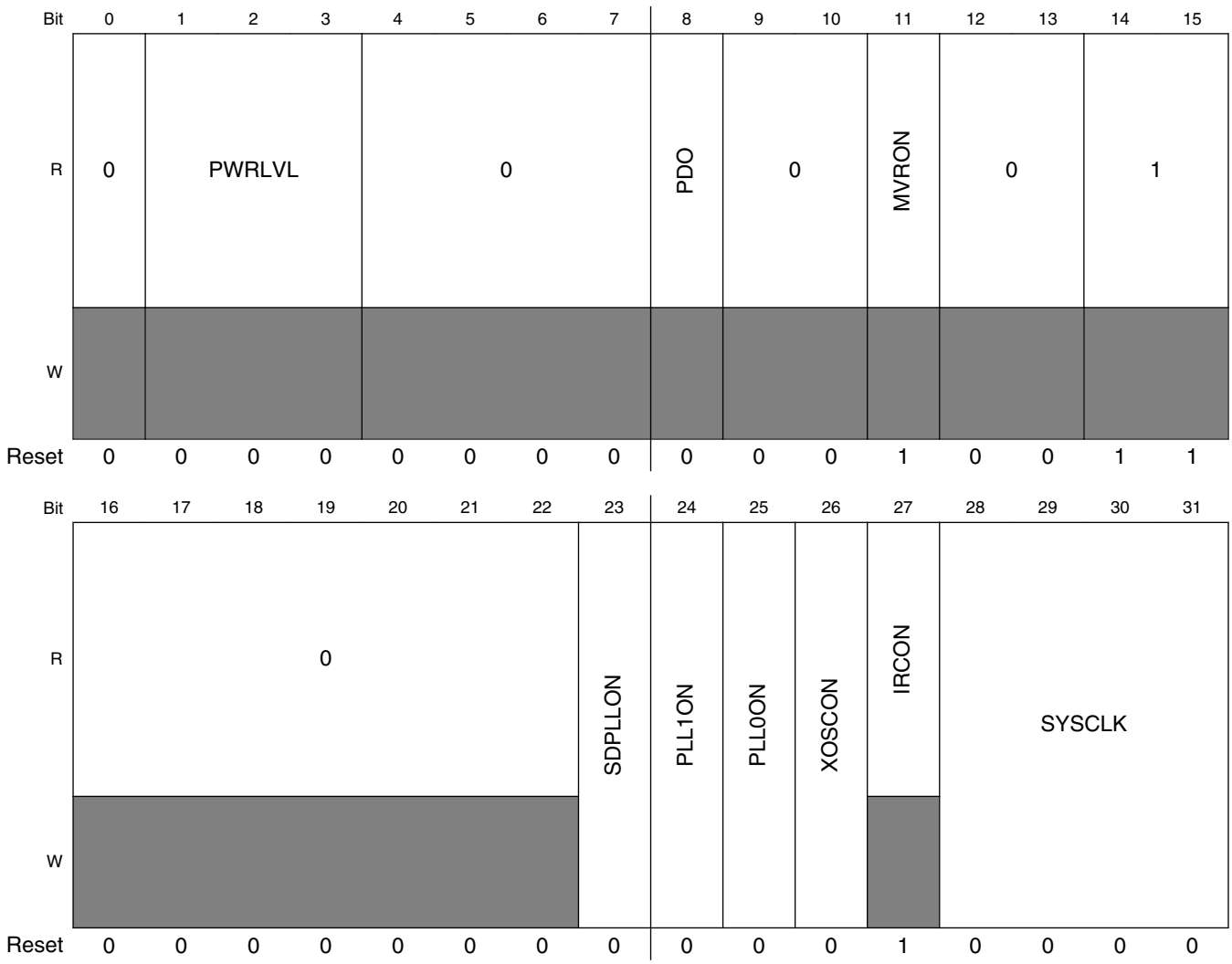
66.3.13 HALT0 Mode Configuration Register (MC_ME_HALT0_MC)

This register configures system behavior during HALT0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 40h offset = 40h



MC_ME_HALT0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off 1 SDPLL is switched on
24 PLL1ON	PLL1 control 0 PLL1 is switched off 1 PLL1 is switched on
25 PLL0ON	PLL0 control 0 PLL0 is switched off 1 PLL0 is switched on
26 XOSCON	XOSC control 0 XOSC is switched off 1 XOSC is switched on
27 IRCON	IRC control 0 IRC is switched off 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
	0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

66.3.14 STOP0 Mode Configuration Register (MC_ME_STOP0_MC)

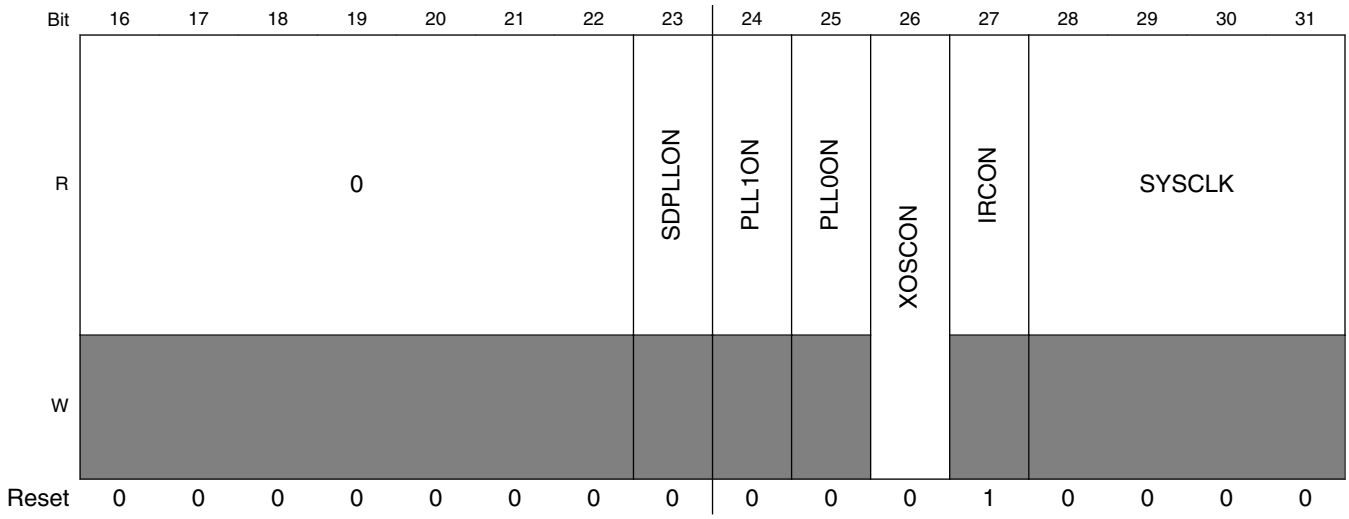
This register configures system behavior during STOP0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_STOP0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDPLLON	SDPLL control 0 SDPLL is switched off 1 SDPLL is switched on
24 PLL1ON	PLL1 control

Table continues on the next page...

MC_ME_STOP0_MC field descriptions (continued)

Field	Description
	0 PLL1 is switched off 1 PLL1 is switched on
25 PLL0ON	PLL0 control 0 PLL0 is switched off 1 PLL0 is switched on
26 XOSCON	XOSC control 0 XOSC is switched off 1 XOSC is switched on
27 IRCON	IRC control 0 IRC is switched off 1 IRC is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 IRC_CLK 0001 XOSC_CLK 0010 PLL0_PHI_CLK 0011 reserved 0100 PLL1_PHI_CLK 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

66.3.15 Peripheral Status Register 0 (MC_ME_PS0)

This register provides the status of the peripherals.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_PIT_RTC_1	S_PIT_RTC_0	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			S_Ethernet_0	S_SIPI_0	0	S_LFAST_0	0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS0 field descriptions

Field	Description
0 S_PIT_RTC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
1 S_PIT_RTC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
2–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 S_Ethernet_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
20 S_SIPI_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_LFAST_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.16 Peripheral Status Register 1 (MC_ME_PS1)

This register provides the status of the peripherals.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			0	0				0	0					S_DTS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									S_CRC_0	0	S_DMAMUX_0	0	0	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 S_DTS	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
15–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_CRC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_DMAMUX_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.17 Peripheral Status Register 2 (MC_ME_PS2)

This register provides the status of the peripherals.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			0	S_LIN_1	0	0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_FLEXCAN_0	S_FLEXCAN_1	S_FLEXCAN_2	0	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS2 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 S_LIN_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_FLEXCAN_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
17 S_FLEXCAN_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
18 S_FLEXCAN_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.18 Peripheral Status Register 3 (MC_ME_PS3)

This register provides the status of the peripherals.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_SAR_ADC_1	0	0	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0			S_FLEXRAY_0	0		0	0	S_IIC_1	0	S_IIC_2	0	S_DSPI_1	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS3 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 S_SAR_ADC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_FLEXRAY_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_IIC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_IIC_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_DSPI_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. **NOTE:** The Bits will only show the status of peripheral being Active after enabling the Module with de-assertion of MDIS bit related to SPI .Enabling the clocks is not enough for this bit. However, this behaviour is only specific to DSPI

Table continues on the next page...

MC_ME_PS3 field descriptions (continued)

Field	Description
	0 Peripheral is frozen 1 Peripheral is active
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.19 Peripheral Status Register 4 (MC_ME_PS4)

This register provides the status of the peripherals.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											S_CRC_1	0	S_DMAMUX_1	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		0	0			S_ETIMER_1	0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

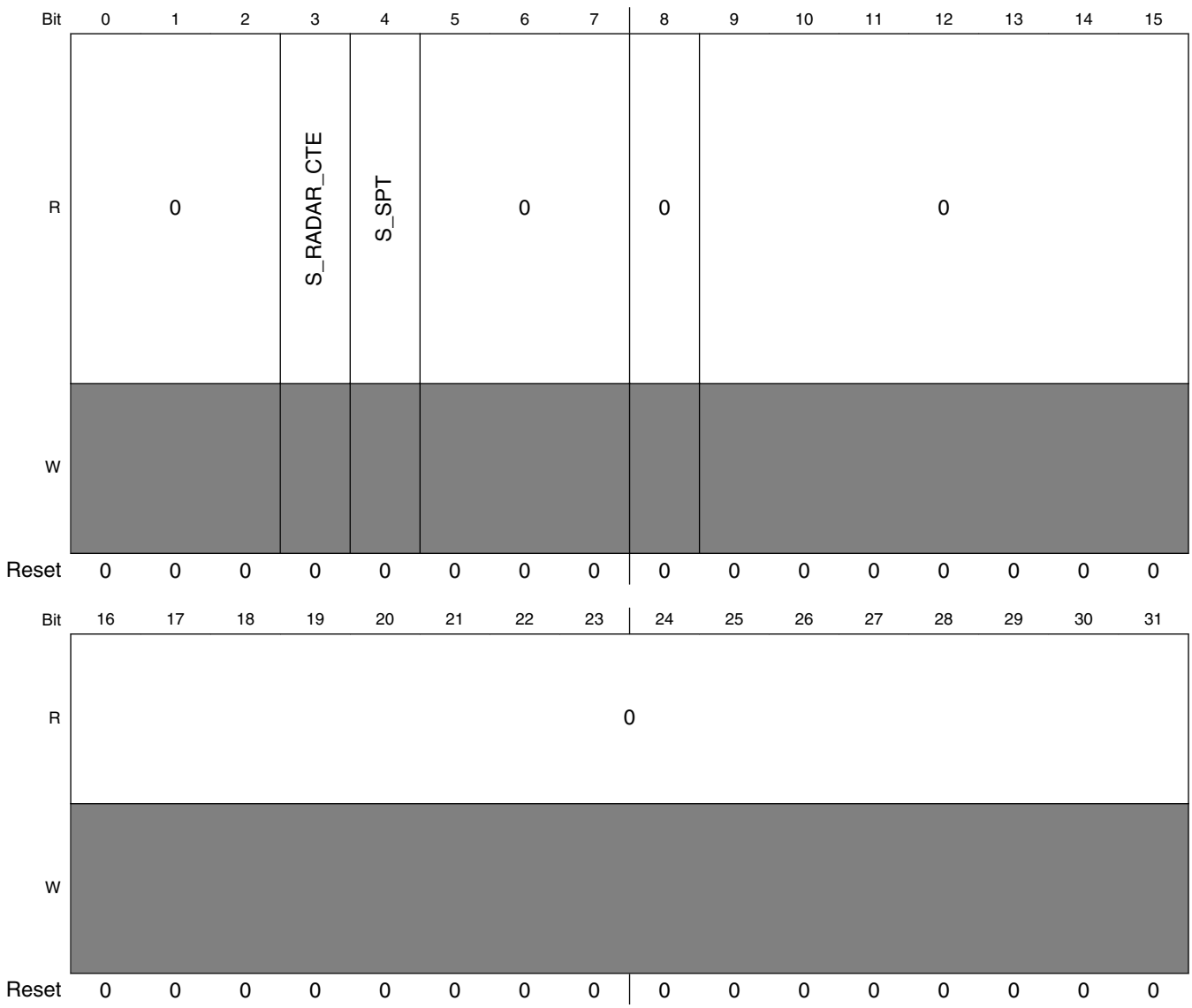
MC_ME_PS4 field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_CRC_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 S_DMAMUX_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_ETIMER_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.20 Peripheral Status Register 5 (MC_ME_PS5)

This register provides the status of the peripherals.

Address: 0h base + 74h offset = 74h



MC_ME_PS5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_RADAR_CTE	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME.

Table continues on the next page...

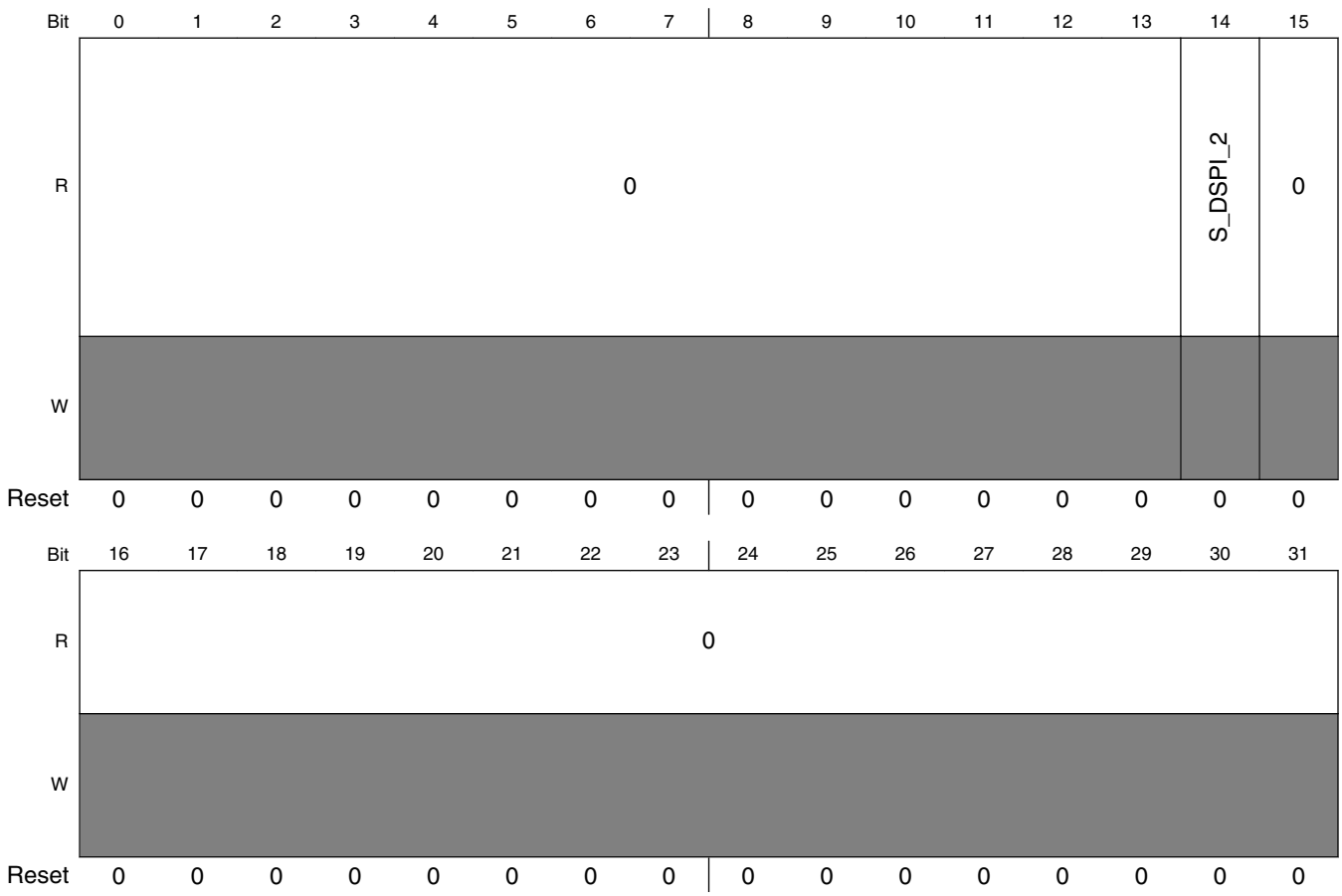
MC_ME_PS5 field descriptions (continued)

Field	Description
	0 Peripheral is frozen 1 Peripheral is active
4 S_SPT	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.21 Peripheral Status Register 6 (MC_ME_PS6)

This register provides the status of the peripherals.

Address: 0h base + 78h offset = 78h



MC_ME_PS6 field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 S_DSPI_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. **NOTE:** The Bits will only show the status of peripheral being Active after enabling the Module with de-assertion of MDIS bit related to SPI .Enabling the clocks is not enough for this bit. However, this behaviour is only specific to DSPI 0 Peripheral is frozen 1 Peripheral is active
15–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.22 Peripheral Status Register 7 (MC_ME_PS7)

This register provides the status of the peripherals.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_FlexPWM_0	0			S_CTU_0	0			0	0	S_ETIMER_2	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	S_WGM_0	S_SAR_ADC_0	0	0	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

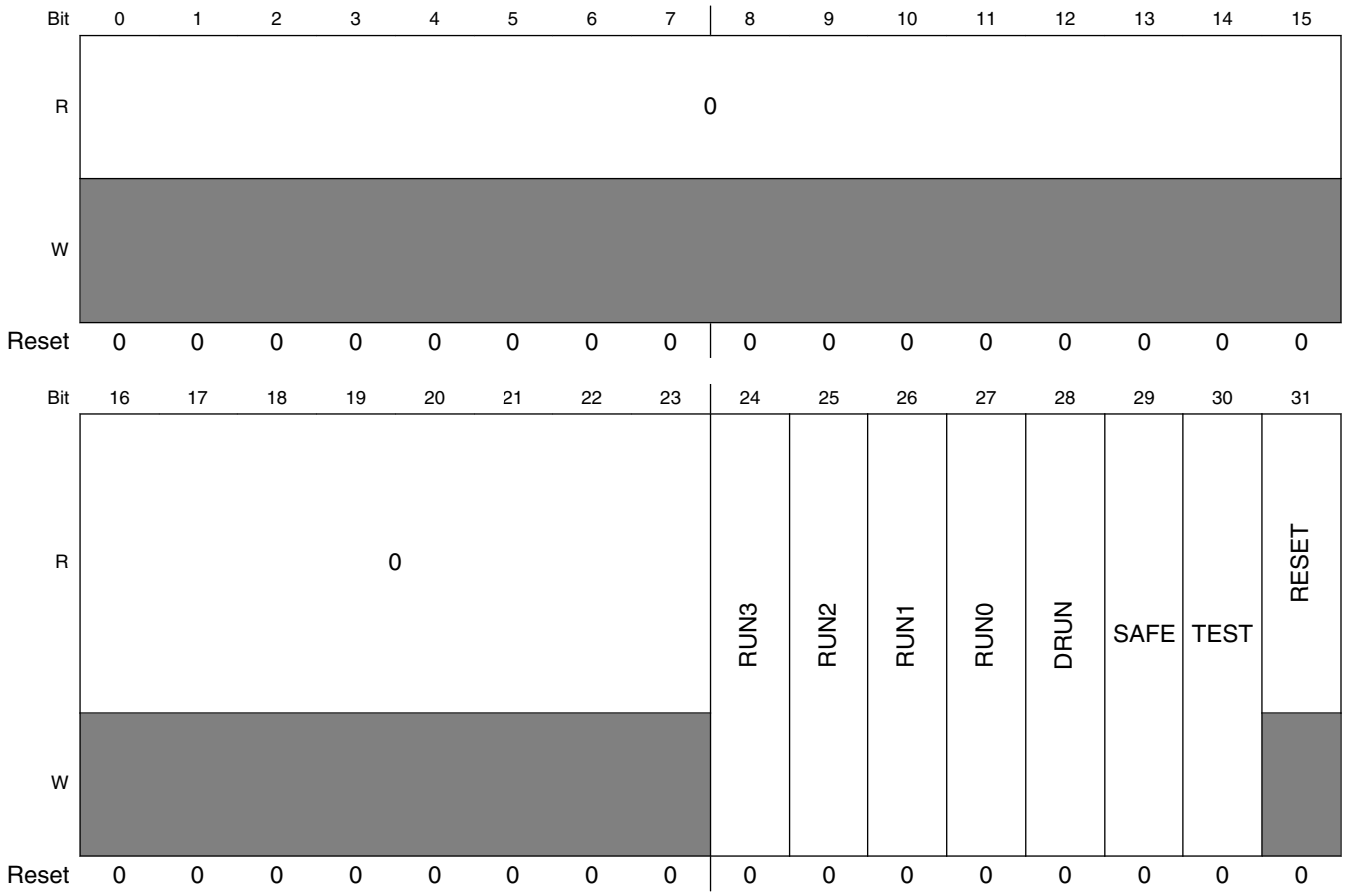
MC_ME_PS7 field descriptions

Field	Description
0 S_FlexPWM_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 S_CTU_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 S_ETIMER_2	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
11–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 S_WGM_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
18 S_SAR_ADC_0	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.23 Run Peripheral Configuration Register (MC_ME_RUN_PC*n*)

These registers configure eight different types of peripheral behavior during run modes.

Address: 0h base + 80h offset + (4d × i), where i=0d to 7d



MC_ME_RUN_PC*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during RUN1

Table continues on the next page...

MC_ME_RUN_PC*n* field descriptions (continued)

Field	Description
	0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

66.3.24 Low-Power Peripheral Configuration Register (MC_ME_LP_PC*n*)

These registers configure eight different types of peripheral behavior during non-run modes.

Address: 0h base + A0h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					STOP0	0	HALT0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_LP_PC*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.25 LFAST_0 Peripheral Control Register (MC_ME_PCTL9)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + C9h offset = C9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL9 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration

Table continues on the next page...

MC_ME_PCTL9 field descriptions (continued)

Field	Description
	011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.26 SIPI_0 Peripheral Control Register (MC_ME_PCTL11)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + CBh offset = CBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL11 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration

Table continues on the next page...

MC_ME_PCTL11 field descriptions (continued)

Field	Description
	001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.27 Ethernet_0 Peripheral Control Register (MC_ME_PCTL12)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL12 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.

Table continues on the next page...

MC_ME_PCTL12 field descriptions (continued)

Field	Description
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.28 PIT_RTC_0 Peripheral Control Register (MC_ME_PCTL30)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL30 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.

Table continues on the next page...

MC_ME_PCTL30 field descriptions (continued)

Field	Description
	0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.29 PIT_RTC_1 Peripheral Control Register (MC_ME_PCTL31)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + DFh offset = DFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL31 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode

Table continues on the next page...

MC_ME_PCTL31 field descriptions (continued)

Field	Description
	NOTE: This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.30 DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)

These registers select the configurations during run and non-run modes for each peripheral.

NOTE

DMA is a platform IP and in low power modes the clock to all platform IPs is shut off. Therefore the DMA cannot be kept ON in low power modes.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL36 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.31 CRC_0 Peripheral Control Register (MC_ME_PCTL38)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL38 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.32 JTAGM Peripheral Control Register (MC_ME_PCTL45)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + EDh offset = EDh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL45 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.33 DTS Peripheral Control Register (MC_ME_PCTL49)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + F1h offset = F1h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL49 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.34 MIPI Peripheral Control Register (MC_ME_PCTL61)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + FDh offset = FDh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL61 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.35 FLEXCAN_2 Peripheral Control Register (MC_ME_PCTL77)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 10Dh offset = 10Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL77 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.36 FLEXCAN_1 Peripheral Control Register (MC_ME_PCTL78)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 10Eh offset = 10Eh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL78 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.37 FLEXCAN_0 Peripheral Control Register (MC_ME_PCTL79)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 10Fh offset = 10Fh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL79 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.38 LINFlex_1 Peripheral Control Register (MC_ME_PCTL91)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 11Bh offset = 11Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL91 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.39 DSPI_1 Peripheral Control Register (MC_ME_PCTL98)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 122h offset = 122h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL98 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.40 IIC_2 Peripheral Control Register (MC_ME_PCTL100)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL100 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.41 IIC_1 Peripheral Control Register (MC_ME_PCTL102)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 126h offset = 126h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL102 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.42 Flexray Peripheral Control Register (MC_ME_PCTL107)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 12Bh offset = 12Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL107 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.43 SAR_ADC_1 Peripheral Control Register (MC_ME_PCTL126)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 13Eh offset = 13Eh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL126 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.44 ETIMER_1 Peripheral Control Register (MC_ME_PCTL137)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 149h offset = 149h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL137 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.45 DMAMUX_1 Peripheral Control Register (MC_ME_PCTL146)

These registers select the configurations during run and non-run modes for each peripheral.

NOTE

DMA is a platform IP and in low power modes the clock to all platform IPs is shut off. Therefore the DMA cannot be kept ON in low power modes.

Address: 0h base + 152h offset = 152h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL146 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.46 CRC_1 Peripheral Control Register (MC_ME_PCTL148)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 154h offset = 154h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL148 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.47 SPT Peripheral Control Register (MC_ME_PCTL187)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 17Bh offset = 17Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL187 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.48 RADAR_CTE Peripheral Control Register (MC_ME_PCTL188)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 17Ch offset = 17Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL188 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.49 DSPI_2 Peripheral Control Register (MC_ME_PCTL209)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 191h offset = 191h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL209 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.50 SAR_ADC_0 Peripheral Control Register (MC_ME_PCTL237)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 1ADh offset = 1ADh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL237 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.51 WGM_0 Peripheral Control Register (MC_ME_PCTL238)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 1AEh offset = 1AEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL238 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.52 ETIMER_2 Peripheral Control Register (MC_ME_PCTL245)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 1B5h offset = 1B5h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL245 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.53 CTU_0 Peripheral Control Register (MC_ME_PCTL251)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 1BBh offset = 1BBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL251 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.54 FlexPWM_0 Peripheral Control Register (MC_ME_PCTL255)

These registers select the configurations during run and non-run modes for each peripheral.

Address: 0h base + 1BFh offset = 1BFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

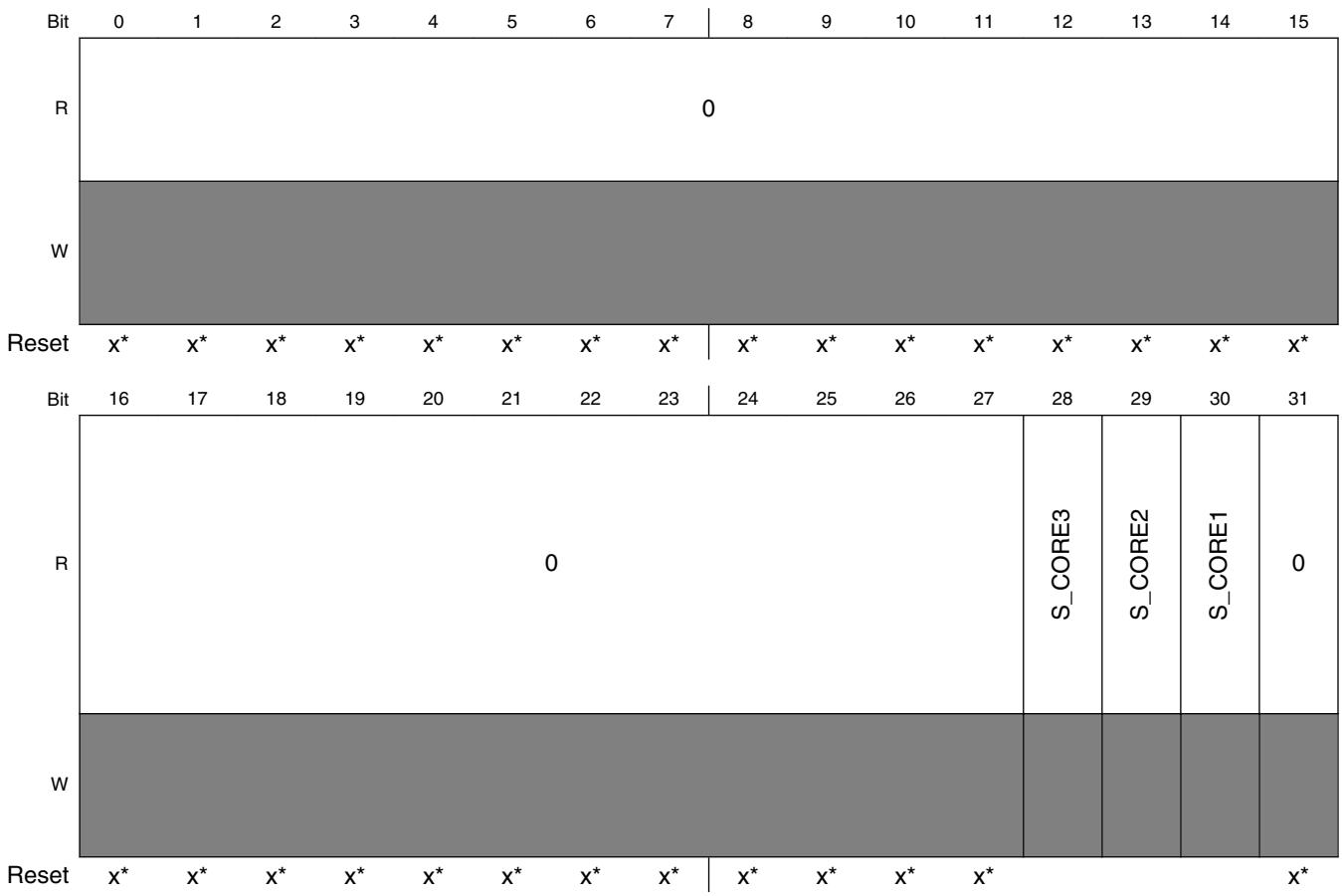
MC_ME_PCTL255 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

66.3.55 Core Status Register (MC_ME_CS)

This register provides the status of each core.

Address: 0h base + 1C0h offset = 1C0h



* Notes:
- Reset value depends on booting core.x = Undefined at reset.

MC_ME_CS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_CORE3	Core 3 status - This bits specifies the current status of z7b which is controlled by the ME_CCTL3 register. 0 z7b is disabled 1 z7b is running
29 S_CORE2	Core 2 status - This bits specifies the current status of z7a which is controlled by the ME_CCTL2 register.

Table continues on the next page...

MC_ME_CS field descriptions (continued)

Field	Description
	0 z7a is disabled 1 z7a is running
30 S_CORE1	Core 1 status - This bits specifies the current status of z4a which is controlled by the ME_CCTL1 register. 0 z4a is disabled 1 z4a is running
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

66.3.56 Core Control Register (MC_ME_CCTL1)

This register controls whether z4a is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

When secondary cores are enabled by using CCTL register in target mode for the first time without setting RMC bit, it starts booting from the BAM location which causes SRAM initialization. If the secondary core is enabled for the first time, the user should always program the CADDR register and set RMC bit before making transition to target mode.

NOTE

Whenever the Z4 core is doing a mode transition from clock enabled to clock enabled mode or from clock disable to clock enabled mode with RMC bit set, the lockstep disablement NCF, NCF[64], will be set. Software should ignore this and take appropriate action to clear it.

Address: 0h base + 1C6h offset = 1C6h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	*	*	*	*	*	*	*	0

* Notes:

- TEST field: reset value loaded from flash during reset sequence
- SAFE field: reset value loaded from flash during reset sequence
- DRUN field: reset value loaded from flash during reset sequence
- RUN0 field: reset value loaded from flash during reset sequence
- RUN1 field: reset value loaded from flash during reset sequence
- RUN2 field: reset value loaded from flash during reset sequence
- RUN3 field: reset value loaded from flash during reset sequence

MC_ME_CCTL1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - z4a is always disabled during STOP0. Always write the reset value to this field
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - z4a is always disabled during HALT0. Always write the reset value to this field
8 RUN3	Core control during RUN3 0 z4a is disabled with clock gated 1 z4a is running
9 RUN2	Core control during RUN2 0 z4a is disabled with clock gated 1 z4a is running
10 RUN1	Core control during RUN1 0 z4a is disabled with clock gated 1 z4a is running
11 RUN0	Core control during RUN0 0 z4a is frozen with clock gated 1 z4a is running
12 DRUN	Core control during DRUN 0 z4a is frozen with clock gated 1 z4a is running
13 SAFE	Core control during SAFE 0 z4a is frozen with clock gated 1 z4a is running
14 TEST	Core control during TEST 0 z4a is frozen with clock gated 1 z4a is running
15 RESET	Core control during RESET - z4a is always disabled during RESET. Always write the reset value to this field

66.3.57 Core Control Register (MC_ME_CCTL2)

This register controls whether z7a is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

When secondary cores are enabled by using CCTL register in target mode for the first time without setting RMC bit, it starts booting from the BAM location which causes SRAM initialization. If the secondary core is enabled for the first time, the user should always program the CADDR register and set RMC bit before making transition to target mode.

Address: 0h base + 1C8h offset = 1C8h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	*	*	*	*	*	*	*	0

* Notes:
- TEST field: reset value loaded from flash during reset sequence
- SAFE field: reset value loaded from flash during reset sequence
- DRUN field: reset value loaded from flash during reset sequence
- RUN0 field: reset value loaded from flash during reset sequence
- RUN1 field: reset value loaded from flash during reset sequence
- RUN2 field: reset value loaded from flash during reset sequence
- RUN3 field: reset value loaded from flash during reset sequence

MC_ME_CCTL2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - z7a is always disabled during STOP0. Always write the reset value to this field
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_CCTL2 field descriptions (continued)

Field	Description
7 HALT0	Core control during HALT0 - z7a is always disabled during HALT0. Always write the reset value to this field
8 RUN3	Core control during RUN3 0 z7a is disabled with clock gated 1 z7a is running
9 RUN2	Core control during RUN2 0 z7a is disabled with clock gated 1 z7a is running
10 RUN1	Core control during RUN1 0 z7a is disabled with clock gated 1 z7a is running
11 RUN0	Core control during RUN0 0 z7a is frozen with clock gated 1 z7a is running
12 DRUN	Core control during DRUN 0 z7a is frozen with clock gated 1 z7a is running
13 SAFE	Core control during SAFE 0 z7a is frozen with clock gated 1 z7a is running
14 TEST	Core control during TEST 0 z7a is frozen with clock gated 1 z7a is running
15 RESET	Core control during RESET - z7a is always disabled during RESET. Always write the reset value to this field

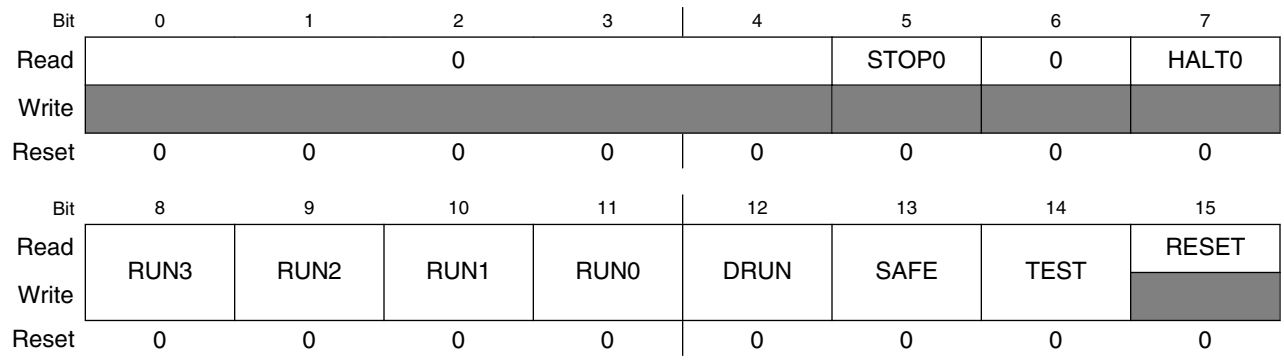
66.3.58 Core Control Register (MC_ME_CCTL3)

This register controls whether z7b is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

When secondary cores are enabled by using CCTL register in target mode for the first time without setting RMC bit, it starts booting from the BAM location which causes SRAM initialization. If the secondary core is enabled for the first time, the user should always program the CADDR register and set RMC bit before making transition to target mode.

Address: 0h base + 1CAh offset = 1CAh



MC_ME_CCTL3 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - z7b is always disabled during STOP0. Always write the reset value to this field
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - z7b is always disabled during HALT0. Always write the reset value to this field
8 RUN3	Core control during RUN3 0 z7b is disabled with clock gated 1 z7b is running
9 RUN2	Core control during RUN2 0 z7b is disabled with clock gated 1 z7b is running
10 RUN1	Core control during RUN1 0 z7b is disabled with clock gated 1 z7b is running
11 RUN0	Core control during RUN0 0 z7b is frozen with clock gated 1 z7b is running
12 DRUN	Core control during DRUN 0 z7b is frozen with clock gated 1 z7b is running

Table continues on the next page...

MC_ME_CCTL3 field descriptions (continued)

Field	Description
13 SAFE	Core control during SAFE 0 z7b is frozen with clock gated 1 z7b is running
14 TEST	Core control during TEST 0 z7b is frozen with clock gated 1 z7b is running
15 RESET	Core control during RESET - z7b is always disabled during RESET. Always write the reset value to this field

66.3.59 Core Control Registers (MC_ME_CADDR1)

This registers gives the boot address for z4a and a bit for controlling whether z4a is to be reset on the next mode change that has z4a configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1E4h offset = 1E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0

* Notes:
- ADDR field: reset value loaded from flash during reset sequence

MC_ME_CADDR1 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by z4a as the boot address (32-bit word aligned) when z4a next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - z4a will be reset on the next mode change that has z4a configured to be running in the target mode.

Table continues on the next page...

MC_ME_CADDR1 field descriptions (continued)

Field	Description
	0 z4a will not be reset on the next mode change 1 z4a will be reset on the next mode change that has z4a configured to be running in the target mode

66.3.60 Core Control Registers (MC_ME_CADDR2)

This registers gives the boot address for z7a and a bit for controlling whether z7a is to be reset on the next mode change that has z7a configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1E8h offset = 1E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0

* Notes:
- ADDR field: reset value loaded from flash during reset sequence

MC_ME_CADDR2 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by z7a as the boot address (32-bit word aligned) when z7a next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - z7a will be reset on the next mode change that has z7a configured to be running in the target mode. 0 z7a will not be reset on the next mode change 1 z7a will be reset on the next mode change that has z7a configured to be running in the target mode

66.3.61 Core Control Registers (MC_ME_CADDR3)

This registers gives the boot address for z7b and a bit for controlling whether z7b is to be reset on the next mode change that has z7b configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Address: 0h base + 1ECh offset = 1ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0

* Notes:
- ADDR field: reset value loaded from flash during reset sequence

MC_ME_CADDR3 field descriptions

Field	Description
0–29 ADDR	Core Address - This field is used by z7b as the boot address (32-bit word aligned) when z7b next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - z7b will be reset on the next mode change that has z7b configured to be running in the target mode. 0 z7b will not be reset on the next mode change 1 z7b will be reset on the next mode change that has z7b configured to be running in the target mode

66.4 Functional Description

66.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Mode Transition Interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0…3 -> RUN0…3, DRUN -> DRUN, SAFE -> SAFE, and TEST -> TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note

It is recommended that software polls the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0 or STOP0 modes.

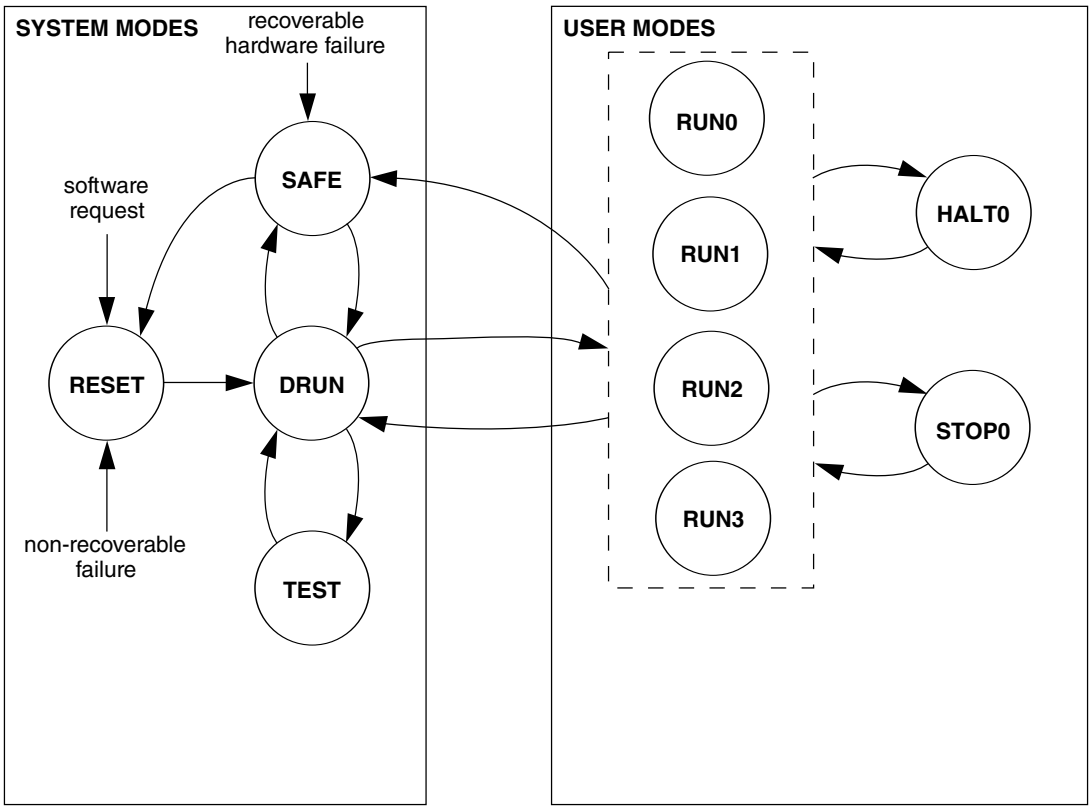


Figure 66-2. MC_ME Mode Diagram

66.4.2 Modes Details

66.4.2.1 RESET Mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0…3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the IRC_CLK is selected as the system clock.

66.4.2.2 DRUN Mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0…3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0011"

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the IRC_CLK as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

66.4.2.3 SAFE Mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with "0010"
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

Note

If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the IRC_CLK is selected as the system clock.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note

If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

66.4.2.4 TEST Mode

The chip enters this mode on the following events:

- from the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0001"

As soon as any of the above events has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to "1111", and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

66.4.2.5 RUN0…3 Modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0…3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0100…0111"
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0…3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0…3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

66.4.2.6 HALT0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1000".

As soon as any of the above events has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen
- the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0…3 mode configuratoin has been fully restored before executing critical code.

66.4.2.7 STOP0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1010".

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the PLL1
- the PLL0
- the SDPLL

If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

This mode can be used to stop all clock sources except IRC. When exiting the STOP0 mode, the IRC clock is selected as the system clock until the target clock is available.

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

NOTE

All interrupts that are masked in INTC should be disabled in the modules as well. Otherwise a hang situation can occur as follows:

- Module issues an interrupt and STOP_ACK to MC_ME is gated.
- MC_ME does not retract and wait for STOP_ACK.
- This interrupt is not available at INTC so the core cannot service it, which causes a hang.

NOTE

A potential system hang scenario may occur if a stop/halt mode request is generated when the interrupts are enabled in active modules and interrupts appears when the MC_ME is transitioning to stop/halt mode.

User needs to adhere to some software guidelines:

- Disable Interrupts before requesting a stop/halt mode change.
- If the previous guideline is not possible ensure any interrupt are finished before stop/halt mode change.
- The wake-up from stop mode should always be done by using external wake-up and not from internal interrupts.

66.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

66.4.3.1 Target Mode Request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Mode Transition Interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

66.4.3.2 Target Mode Configuration Loading

On completion of the Target Mode Request step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in the following table. A ' √' indicates that a given resource is configurable for a given mode.

Table 66-2. MC_ME Resource Control Overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
IRC		√					
	on	on	on	on	on	on	on
XOSC		√		√	√	√	√
	off	off	off	off	off	off	off
PLL0		√		√	√	√	
	off	off	off	off	off	off	off

Table continues on the next page...

Table 66-2. MC_ME Resource Control Overview (continued)

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
SDPLL		√		√	√	√	
	off	off	off	off	off	off	off
PLL1		√		√	√	√	
	off	off	off	off	off	off	off
MVREG	on	on	on	on	on	on	on
PDO		√	√				√
	off	off	on	off	off	off	off

66.4.3.3 Peripheral Clocks Disable

On completion of the Target Mode Request step and the system clock frequency ramp-down of the System Clock Switching step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0…7 and ME_LP_PC0…7, and the peripheral control registers ME_PCTLn

Note

The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Peripheral Clock Gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

66.4.3.4 Processor Low-Power Mode Entry

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the HALT0,STOP0 mode, the MC_ME requests each processor to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the ME_CCTLn registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

66.4.3.5 Processor Clocks Disable

If, on completion of the Processor Low-Power Mode Entry step, and all processors which are configured in the ME_CCTLn registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

warning

Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

66.4.3.6 System Clock Disable

If, on completion of the Processor Clocks Disable step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

66.4.3.7 Clock Sources Switch-On

On completion of the Processor Low-Power Mode Entry step, the MC_ME switches on all clock sources based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the IRC
- the XOSC
- the PLL1
- the PLL0
- the SDPLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

66.4.3.8 Pad Outputs-On

On completion of the step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- the slew rate control mechanism is switched on

66.4.3.9 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed, and if the value of the PWRLVL field of the ME_<target mode>_MC register is process is completed.

66.4.3.10 System and Processor Clocks Enable

The MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the ME_CCTLn registers to be running in the target mode and which have their RMC bits in the ME_CADDRn registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the S_MTRANS bit of the ME_GS register at which time they are deasserted.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

66.4.3.11 Processor Low-Power Mode Exit

On completion of the System and Processor Clocks Enable step, the MC_ME requests all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

66.4.3.12 System Clock Switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the IRC_CLK takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the IRC has stabilized).

- The target clock configuration for the XOSC_CLK takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (i.e., the XOSC has stabilized).
- The target clock configuration for the PLL1_PHI_CLK takes effect only after the S_PLL1 bit of the ME_GS register is set by hardware (i.e., the PLL1 has stabilized).
- The target clock configuration for the PLL0_PHI_CLK takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (i.e., the PLL0 has stabilized)
- If the clock is to be disabled, the SYSCLK bit field should be programmed with "1111". This is possible only in theTEST mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register not equal to that of the ME_<current mode>_MC register's, a progresive clock switching is performed.

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Clock Sources Switch-On process has completed if the target system clock source is one of the following:
  - the IRC
  - the XOSC
  - the PLL1
  - the PLL0
- the Peripheral Clocks Disable process has completed in order not to change the system clock frequency before peripherals close their internal activities
- the Peripheral Clocks Enable process has completed

An overview of system clock source selection possibilities for each mode is shown in the following table. A '√' indicates that a given clock source is selectable for a given mode.

Table 66-3. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
IRC_CLK	√	√	√	√	√	√	√

Table continues on the next page...

Table 66-3. MC_ME System Clock Selection Overview (continued)

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
	(default)	(default)	(default)	(default)	(default)	(default)	(default)
XOSC_CLK		√		√	√	√	
PLL1_PHI_CLK		√		√	√	√	
PLL0_PHI_CLK		√		√	√	√	
system clock is disabled		√[1]					

1. disabling the system clock during TEST mode will require a reset in order to exit TEST mode

66.4.3.13 Pad Switch-Off

If the PDO bit of the ME_<target mode>_MC register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

This step is executed only after the Peripheral Clocks Disable process has completed.

66.4.3.14 Clock Sources Switch-Off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources switched off at this step:

- the IRC
- the XOSC
- the PLL1

- the PLL0
- the SDPLL

For the clock sources related to the sytem clock, this step is executed only after the System Clock Switching process has completed.

66.4.3.15 Current Mode Update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

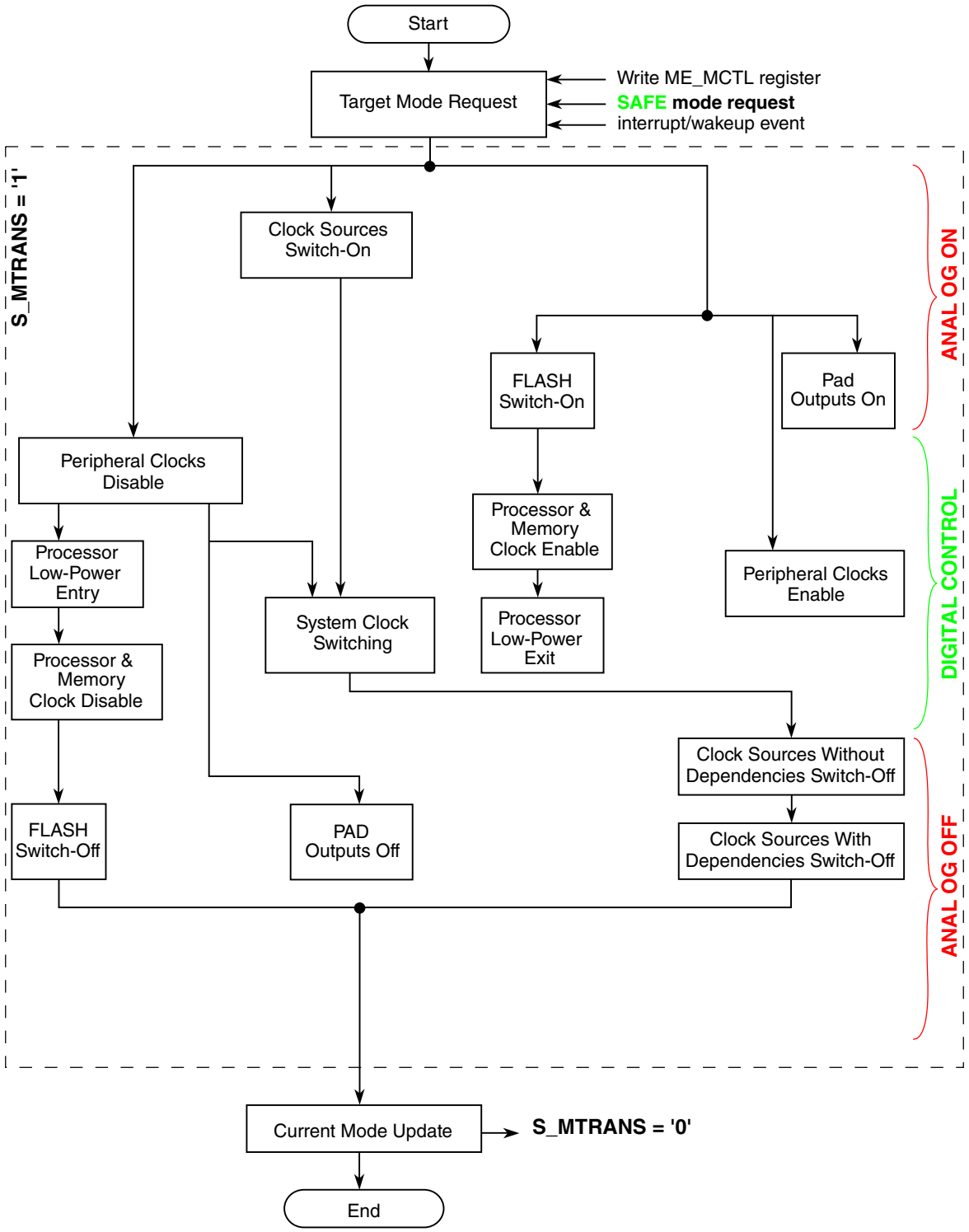


Figure 66-3. MC_ME Transition Diagram

66.4.4 Mode transition and progressive clock switching

The MC_ME initiates a progressive clock switching via MC_CGM whenever the **ME_<current mode>_MC.PWRLVL** field is different from **ME_<target mode>_MC.PWRLVL** bit field. Broadly, the mode transition process can be divided into three steps with respect to progressive cclock switching - RAMP DOWN, LOAD CHANGE and RAMP UP. One or more of these steps may not be needed based on the current and target mode configirations. The clock switching (from current system clock source to target system clock source) can happen before or after the LOAD CHANGE step.

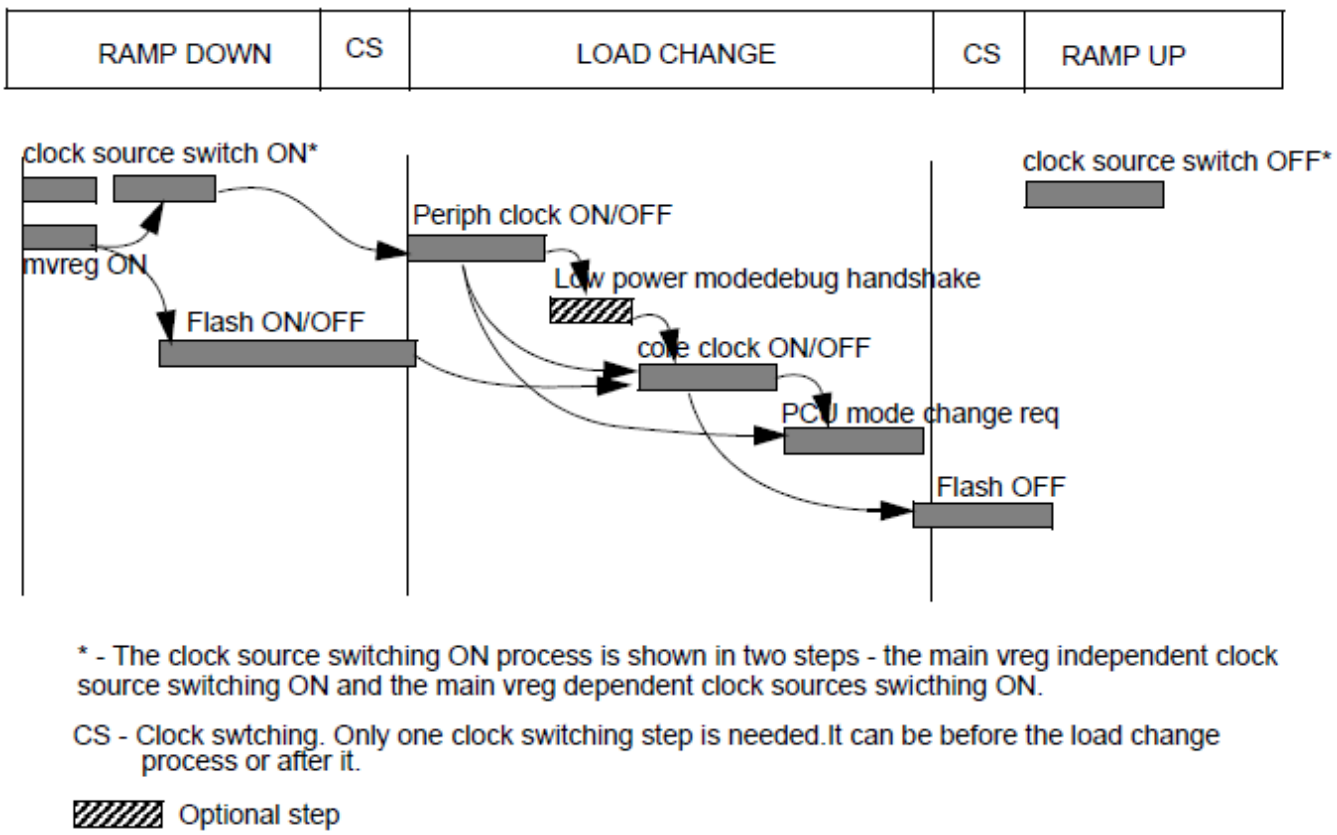


Figure 66-4. MC_ME mode transition process with PCS

The table below shows the different ways the mode transition process can complete.

Table 66-4. Mode transition steps

Pwrlvl	Source clock	Target clock	Mode transition steps
Same	IRC	Non-IRC	LoadChange -> ClockSwitch
	IRC	IRC	LoadChange
	Non-IRC	IRC	LoadChange -> ClockSwitch

Table continues on the next page...

Table 66-4. Mode transition steps (continued)

Pwrlvl	Source clock	Target clock	Mode transition steps
	Non-IRC	Non-IRC	LoadChange -> ClockSwitch
Different	IRC	Non-IRC	LoadChange -> ClockSwitch -> RampUp
	IRC	IRC	LoadChange
	Non-IRC	IRC	RampDown -> ClockSwitch -> LoadChange
	Non-IRC	Non-IRC	RampDown -> ClockSwitch -> LoadChange -> RampUp

66.4.5 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the IRC_CLK is selected as the system clock, IRC must be on.
- If the XOSC_CLK clock is selected as the system clock, XOSC must be on.
- If the PLL1_PHI_CLK clock is selected as the system clock, PLL must be on.
- If the PLL0_PHI_CLK clock is selected as the system clock, PLL0 must be on.

Note

Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration "00" for the FLAON bit field is reserved.
- System clock configurations marked as 'reserved' may not be selected.
- Configuration "1111" for the SYSCLK bit field is allowed only for theTEST mode, and only in this case may all system clock sources be turned off.

warning

If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

66.4.6 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

66.4.6.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the Protection of Mode Configuration Registers, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is '1'.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0…7/ME_LP_PC0…7 and ME_PCTLn registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is '1'.

If an attempt to write to ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is '1'.

NOTE

In case of illegal clock configuration in low power mode, low power mode entry hangs. The software needs to take care that peripherals enabled in low power mode have their clocks enabled in it.

66.4.6.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.
- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occurs while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA and S_MRI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.

- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0…3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.
- Self-transition requests (e.g., RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (i.e., S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0…3 mode properly (i.e., all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

66.4.6.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1' .

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (i.e., programming of ME_MCTL register).

66.4.6.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

66.4.7 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0…7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0…3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTLn registers.

The low-power peripheral configuration registers ME_LP_PC0…7 are chosen only during the low-power modes HALT0 and STOP0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTLn registers.

Any modifications to the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTLn register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

66.4.8 Application Example

The following figure shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

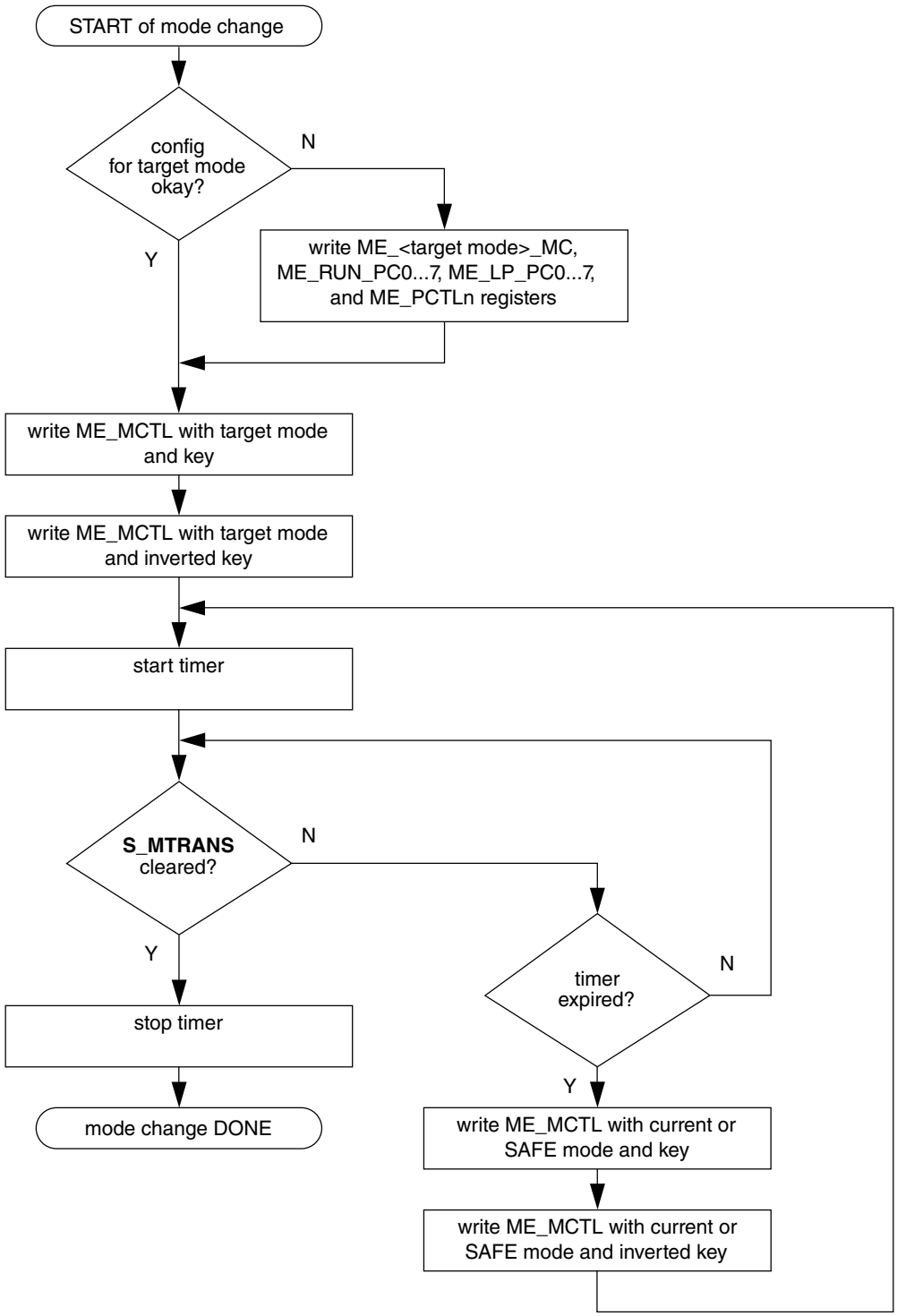


Figure 66-5. MC_ME Application Example Flow Diagram

Chapter 67
Nexus Crossbar Multi-Master Client (NXMC)

67.1 Chip-specific NXMC information

67.1.1 NXMC block connections

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message.

The following figure shows a block diagram illustrating the NXMC integration.

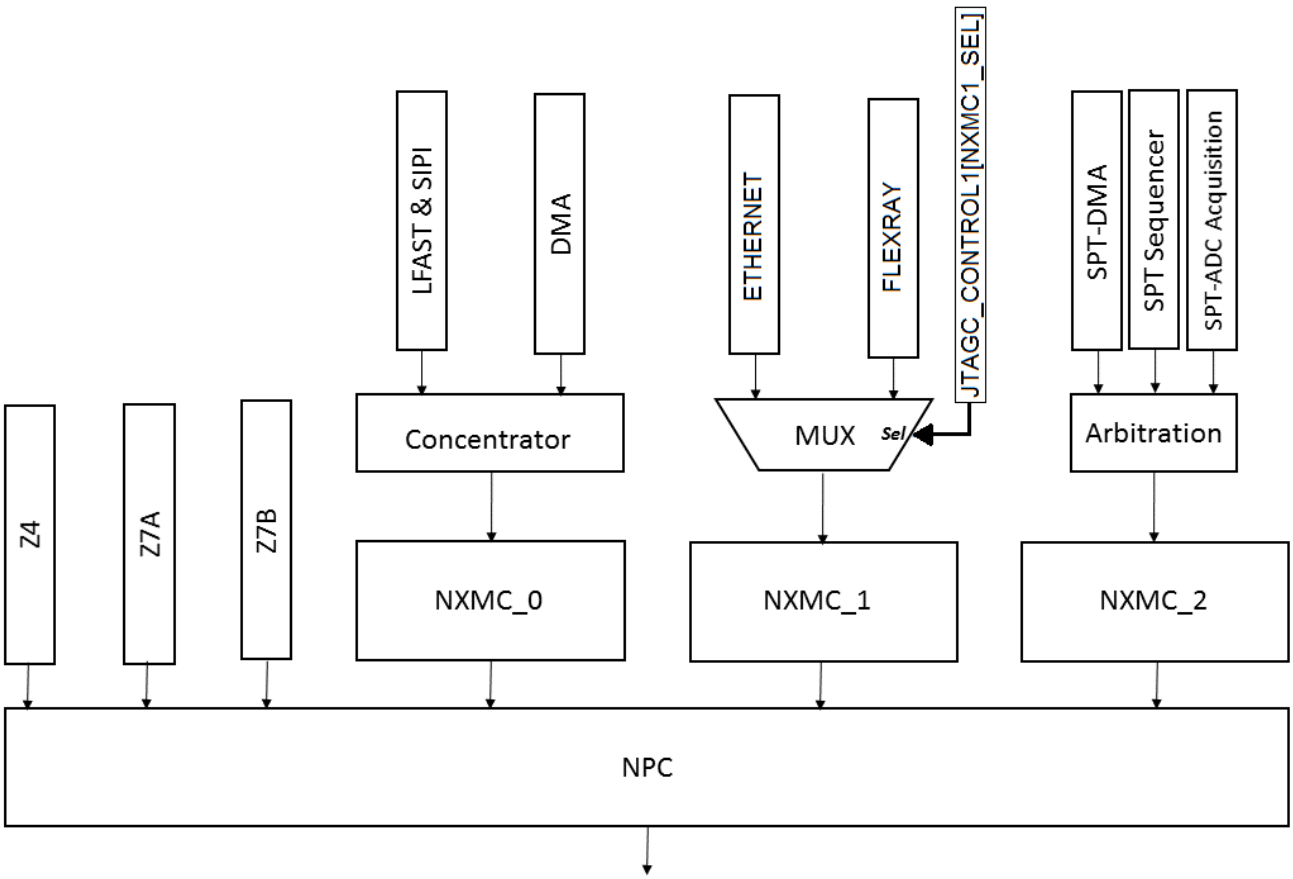


Figure 67-1. NXMC integration block diagram

The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 67-1. NXMC source and master client IDs

Nexus Client	Trace Master	Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
NXMC0	DMA	0b1100	0b0010
	SIPI/LFAST		0b0100
NXMC1	FlexRay	0b1101	0b0011
	Ethernet		0b0101
NXMC2	SPT-DMA (PDMA)	0b1110	0b0110
	SPT-MIPICSI2 Acquisition (bypass SDMA)		0b1010[1]
	SPT-Sequencer (Command Sequencer DMA)		0b1101
	SPT-ADC/MIPICSI2-Acquisition(Main SDMA)		0b1110[1]

1. See SPT_SDMA_CTRL1[HMASTER_ENC_DIS] for control of Nexus trace MASTER ID for SPT-ADC-Acquisition (SDMA). This register field controls the trace master id contain in the MID field of the Nexus message. The NXMC2 either sends the SPT-ADC-Acquisition ID or it sends an SD ADC identifier depending on the HMASTER_ENC_DIS setting.

67.1.2 Avoiding trace overflows

For preventing the trace overflows the following configurations can be programmed.

- DMA – FIFO watermark level(MCB_NPC_SPECIAL_ENABLE.WATERMARK[3:0]), NPC NAL FIFO EXTENSION_MODE, DMA Bandwidth Control (DMA_TCDn_CSR.BWC[1:0]).
- SPT – FIFO watermark level, NPC NAL FIFO EXTENSION_MODE, SPT Throttle Control (MCB_MISC2.SPT_NEXUS_THROTTLE_CONTROL) and NXMC_2_FIFO_EN.
- All other masters - FIFO watermark level, NPC NAL FIFO EXTENSION_MODE.

The FIFO watermark level can be used to throttle trace from various nexus sources inside chip. Its default value is 0x3 and can be programmed maximum upto 0x7. On increasing the value from 0x3 to 0x7, it is seen that the fifo-full error messages in traces are decreased.

In the NPC NAL FIFO EXTENSION_MODE, the FIFO mentioned above can be extended to be 0xF deep. Please follow the guidelines in register description of MCB_NPC_SPECIAL_ENABLE[NPC_NAL_FIFO_EXTENSION_MODE], when using this mode. Trace overflows will be further reduced when using this mode.

For DMA, in the TCD, a BWC field can be programmed, which puts empty cycles between DMA transfers, thereby reducing the throughput and chances of trace overflows.

For SPT, SPT_NEXUS_THROTTLE_CONTROL can be programmed. Making this value ‘0’, will cause SPT to give full-performance. Increasing this value, will decrease the SPT traffic throughput. So, the NXMC-fifo-overflows in case of SPT can also be managed by this configuration. In addition to this, the SPT also takes NXMC-fifo-watermark-level into consideration. SPT traffic is stalled when this watermark level crosses half-full threshold. SPT traffic restarts once the watermark level comes below the half-full threshold mark. Since the SPT is taking NXMC-fifo-watermark-level into consideration, any stall in the trace path will stall the SPT traffic as well. If 'Aurora clock' limitation documented in System clock frequency limitations is not met ,then NAL-fifo stall will happen which will cause NXMC2-fifo to stall and that in turn will cause SPT operation to stall.

While tracing SPT, NXMC_2_FIFO can be enabled by programming MCB_MISC2[NXMC_2_FIFO_EN]. This FIFO between SPT and NXMC can handle intermittent bursts of data, but if the amount of data traced is very large (for eg, if both CSDMA and PDMA traces are enabled at full throughput) then this FIFO will show overflow (MCB_NEX_FIFO_STATUS[31])

67.1.3 Nexus Message Timestamp

The Nexus message TSTAMP field is a timestamp value provided by a counter clocked by SYS_CLK. Trace messages generated by the e200z420 core and e200z760 cores capture a 30-bit timestamp from the counter. Trace masters that interface through a Nexus Crossbar Multi-Master Client module (LFAST/SIPI, DMA, ENET, FlexRay, SPT-PDMA, SPT Command Sequencer, SPT-SDMA) capture a 16-bit timestamp value from this counter. Thus, the timestamp for these trace masters will roll-over to zero before the timestamps from e200zx core trace masters.

67.2 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.

67.2.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on internal triggers.
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin

- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

67.3 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in Table 67-3.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC. The following table describes the JTAG pins.

Table 67-2. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 67-3. NXMC Auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: • one of the internal or external watchpoints has occurred and EOC = 2'b00 • system-level debug mode was entered (ipg_debug) and EOC = 2'b01

Table continues on the next page...

Table 67-3. NXMC Auxiliary pins (continued)

Auxiliary port	Input/ output	Description
		$\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

67.4 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 67-4. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 67-5)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 67-5)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 67-6)

Table continues on the next page...

Table 67-4. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
	14	14	ECODE	fixed	Error code (See Table 67-7)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 67-5)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 67-5)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 67-23)

Table 67-5. Data trace size (DSZ) encodings (TCODE = 58, 59, 60, 61)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 67-6. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost

Table continues on the next page...

Table 67-6. Error Type (ETYPE) encodings (applicable to all error messages) (continued)

ETYPE	Description
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 67-7. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	Reserved
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

67.5 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete list of registers is shown in this table.

Table 67-8. NXMC registers

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Development Control 1 (DC1)	2	R/W	04	05
Development Control 2 (DC2)	3	R/W	06	07
Watchpoint Trigger (WT)	B	R/W	16	17
Data Trace Control (DTC)	D	R/W	1A	1B
Data Trace Start Address1 (DTSA1)	E	R/W	1C	1D
Data Trace Start Address2 (DTSA2)	F	R/W	1E	1F
Data Trace End Address1 (DTEA1)	12	R/W	24	25
Data Trace End Address2 (DTEA2)	13	R/W	26	27
Breakpoint/Watchpoint Control Register1 (BWC1)	16	R/W	2C	2D
Breakpoint/Watchpoint Control Register2 (BWC2)	17	R/W	2E	2F
Breakpoint/Watchpoint Address Register1 (BWA1)	1E	R/W	3C	3D
Breakpoint/Watchpoint Address Register2 (BWA2)	1F	R/W	3E	3F

67.5.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EOC		0	0	WEN	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	EIC		TM		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC1 is described in this table.

Table 67-9. DC1 register field description

Field	Description
31:29	Reserved for future functionality (Read/Write – default values must not be changed for normal operation)
28:27 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
26:25	Reserved for future functionality (read as 0)
24	Watchpoint Trace Enable.

Table continues on the next page...

Table 67-9. DC1 register field description (continued)

Field	Description
WEN	0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23:5	Reserved for future functionality (read as 0)
4:3 EIC	$\overline{\text{EVTI}}$ Control. 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
2:0 TM	Trace Mode. 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

67.5.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in this table.

Table 67-10. DC2 register field description

Field	Description	
31:24 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration[1]. 00000000 1XXXXXXX X1XXXXXX	No Watchpoints trigger $\overline{\text{EVTO}}$ Reserved (This is read/write field, so default value must not be changed for normal operation) Reserved (This is read/write field, so default value must not be changed for normal operation)

Table continues on the next page...

Table 67-10. DC2 register field description (continued)

Field	Description	
	XX1XXXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXX1XXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXXX1XXX	Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$
	XXXXX1XX	Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XXXXXX1X	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXXXXXX1	Reserved (This is read/write field, so default value must not be changed for normal operation)
23:1	Reserved for future functionality (read as 0)	
0	Reserved for future functionality (Read/Write – default values must not be changed for normal operation)	

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.

67.5.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable.

The WT bits can be used to produce an address related window for triggering trace messages. This table shows the format of the WT register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The WT register is described in this table.

Table 67-11. WT register field description

Field	Description[1]
31:26	Reserved for future functionality (read as 0)
25:23 DTS	Data Trace Start Control. 000 Trigger disabled 001 Reserved 010 Reserved 011 Reserved

Table continues on the next page...

Table 67-11. WT register field description (continued)

Field	Description[1]
	100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
22:20 DTE	Data Trace End Control. 000 Trigger disabled 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
19:0	Reserved for future functionality (read as 0)

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

67.5.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. This table shows the format of the DTC register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RWT1		RWT2		0	0	0	ME	0	0			MID				0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in this table.

Table 67-12. DTC field description

Field	Description
31:30 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace

Table continues on the next page...

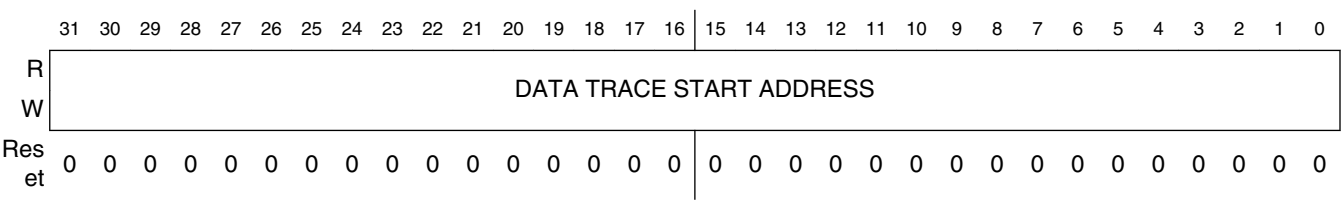
Table 67-12. DTC field description (continued)

Field	Description
29:28 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
27:25	Reserved for future functionality
24 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
23:20	Reserved for future functionality
19:16 MID	ID of the system bus Master to be traced[1]
15:8	Reserved for future functionality
7 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

67.5.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. This table shows the format of the DTSA1 and DTSA2 registers.



67.5.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. This table shows the format of the DTEA1 and DTEA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA TRACE END ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 67-13. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note

DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

67.5.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. This table shows the format of the BWC1 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC1 register is described in this table.

Table 67-14. BWC1 field description

Field	Description	
31:30	Breakpoint/Watchpoint #1 Enable.	
BWE1	00	Internal Nexus Watchpoint #1 disabled
	01	Reserved
	10	Reserved
	11	Internal Nexus Watchpoint #1 enabled
29:28	Breakpoint/Watchpoint #1 Read/Write Select.	
BRW1	00	Watchpoint #1 hit on read accesses
	01	Watchpoint #1 hit on write accesses
	10	Watchpoint #1 on read or write accesses
	11	Reserved
27:18	Reserved for future functionality (read as 0)	
17:16	Breakpoint/Watchpoint #1 Register Compare.	
BWR1	00	No Register Compare (same as BWC1[31:30] = 2'b00)
	01	Reserved
	10	Compare with BWA1 value
	11	Reserved
15	Breakpoint/Watchpoint #1 Type.	
BWT1	0	Reserved
	1	Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)	

67.5.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. This table shows the format of the BWC2 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2		BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC2 register is described in this table.

Table 67-15. BWC2 field description

Field	Description
31:30 BWE2	Breakpoint/Watchpoint #2 Enable. 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
29:28 BRW2	Breakpoint/Watchpoint #2 Read/Write Select. 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type. 0 Reserved 1 Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)

67.5.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. This table shows the format of the BWA1 and BWA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

67.5.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 67-8. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

67.6 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion. The JTAG ID register should be read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

67.6.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

67.6.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

67.6.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

67.6.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 67-8.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 67-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

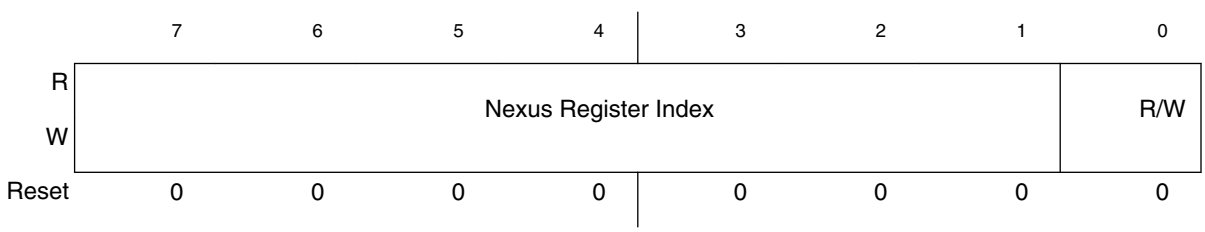


Table 67-16. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 67-8
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.

a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.

b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

67.7 Functional description

The following sections describe the functionality of the NXMC module.

67.7.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

67.7.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

67.7.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

67.7.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32 , 64. Data write message and data read message information is messaged out in the format shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 67-17. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

67.7.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

67.7.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in this figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 67-18. DTM error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

NOTE

The internal queue FIFO is very small. This causes FIFO overflows whenever Trace is enabled on any of the AHB clients and the AHB master module generates >4 burst accesses. If the customer is experiencing a lot of overflow error messages the amount of data snooped by the NXMC should be reduced.

67.7.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 67-20):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message. This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in the following figure. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 67-19. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in this table.

Table 67-20. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

67.7.1.3 Enabling data trace messaging operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).

- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

67.7.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > DTM).

67.7.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in this figure.

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 67-2. Relative addressing example

67.7.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

67.7.1.7 System bus cycle special cases

The system bus cycle special cases are shown in this table.

Table 67-21. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

67.7.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in Data write and data read messages for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

67.7.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

67.7.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in System bus multi-master tracing).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in Table 67-22 and Table 67-23.

Table 67-22. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 67-23. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1[1]	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

67.7.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in this figure.

Table 67-24. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

Chapter 68
Nexus Port Controller (NPC)

68.1 Chip-specific NPC information

68.1.1 EVTO sharing - Pulse time

In this device, whenever a Nexus client drives EVTO input of NPC for one SYS_CLK period, the NPC drives EVTO on pad for one SYS_CLK period on the following clock.

68.1.2 Trace messages for transactions with error responses

For transactions of trace clients which result in error responses, traces may or may not be generated. Such transactions will be reported to MEMU.

The Nexus clients in this device do not perform speculative execution. When speculative execution is done, a trace message may be output before it is known whether the branch was actually taken. Since the Nexus clients in this device do not support it, the CANCEL field is not present in the trace messages of this device.

68.1.3 Trace loss during resets

NOTE

Whenever reset occurs (reset which breaks debugger trace connection) and nexus tracing is ON, debug trace data stored in internal fifo corresponding to instructions executed before reset instruction will be lost.

68.1.4 EVTI_IN pin

EVTI_IN is an active low signal. The NPC Port Configuration Register's EVT_EN bit configures the EVTI_IN and EVTO_B pins for EVTI/EVTO port functionality. If it is chosen by the user to not use this configuration and to use the SIUL_MSCR configuration register for this purpose, then it is recommended that the user also sets the PUE and PUS bits to 1 along with setting the IBE and SSS configurations. Otherwise, a 0 event may be seen in EVTI_IN which will trigger the EVTI functions in chip.

68.2 Introduction

The following figure is a block diagram of the Nexus Port Controller (NPC) block.

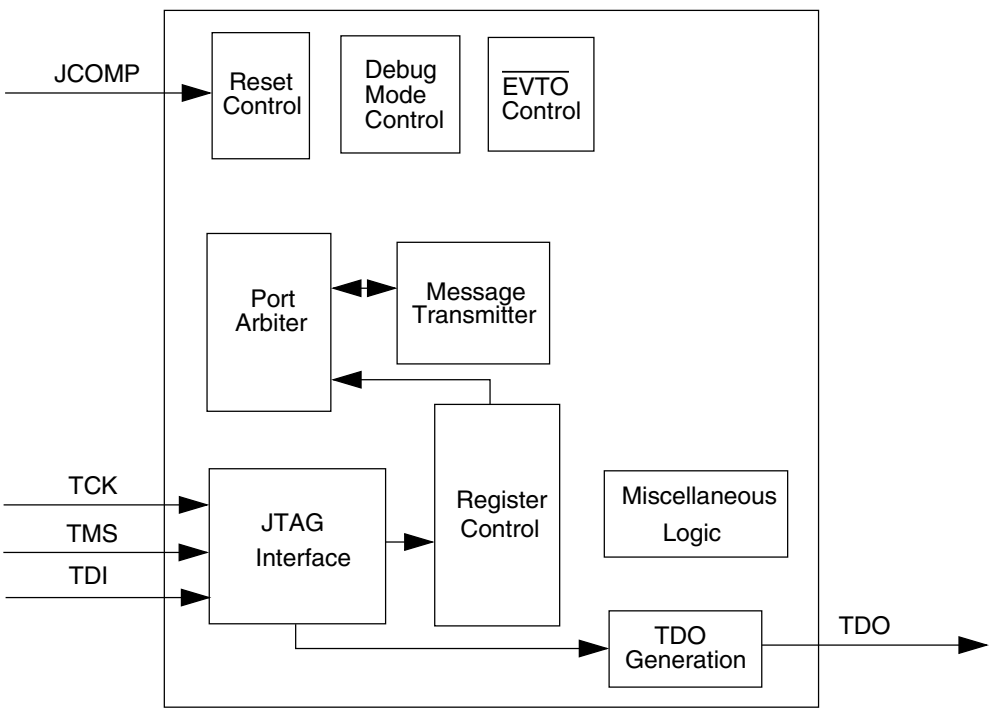


Figure 68-1. Nexus Port Controller Block Diagram

68.2.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2001 standard. The blocks share input and output ports that interface with the

development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

68.2.2 Features

The NPC block performs the following functions:

- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on JCOMP input, and power-on reset status
- Selectable mode for transmission of processor status information on MDO Auxiliary Output Port pins
- Controls arbitration for ownership of the Nexus Aurora Link (NAL) in Aurora Link mode

68.2.3 Modes of operation

The NPC block uses the JCOMP input, and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

68.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, JCOMP is not set for Nexus access or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset or JCOMP not set for NPC operation only
- Registers default back to their reset values

68.2.3.2 Disabled-Port Mode

In disabled-port mode, output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

68.2.3.3 Aurora Link Mode

In Aurora Link mode, trace data is transmitted through the Nexus Aurora Link (NAL). For SoCs that support a Nexus Aurora Link, the FPM bit in the PCR functions as the Aurora Link enable. Aurora Link Mode is enabled by setting the MCKO_EN and FPM bits and selecting a non-zero value for the ALC_NUM_LN bits in the PCR. The MCKO_DIV value should be set to select a clock frequency equal to the system clock frequency (MCKO_DIV = 000).

68.3 External signal description

68.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in the following table. Pull information for the JTAG signals can be found in the JTAG chapter.

Table 68-1. NPC Signal Properties

Name	Port	Function	Reset State
EVTO_B	Nexus	Event Out pin	0b1
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	—

Table continues on the next page...

Table 68-1. NPC Signal Properties (continued)

Name	Port	Function	Reset State
TCK	JTAG	Test Clock Input	—
TDI	JTAG	Test Data Input	—
TDO	JTAG	Test Data Output	High Z[1]
TMS	JTAG	Test Mode Select Input	—

1. TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

68.3.2 Detailed signal descriptions

This section describes each of the signals listed in Table 68-1 in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

68.3.2.1 EVTO_B — Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

68.3.2.2 JCOMP — JTAG Compliancy

The JCOMP signal provides the ability to share the TAP. The NPC TAP controller remains in reset until it is enabled using the JCOMP signal.

68.3.2.3 TCK — Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

68.3.2.4 TDI — Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

68.3.2.5 TDO — Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

68.3.2.6 TMS — Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

68.4 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

The following table shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers.

Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 68-2. NPC registers

Index	Register
0	Device ID Register (DID)
126	Aurora Configuration Register (ACR)
127	Port Configuration Register (PCR)

68.4.1 Register descriptions

This section consists of NPC register descriptions.

68.4.1.1 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

68.4.1.2 Instruction register

The NPC block uses a 4-bit instruction register as shown in the following table. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

68.4.1.3 Nexus Device ID (DID) Register

The device identification register, shown in the following figure, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the output port.

Register index: 0

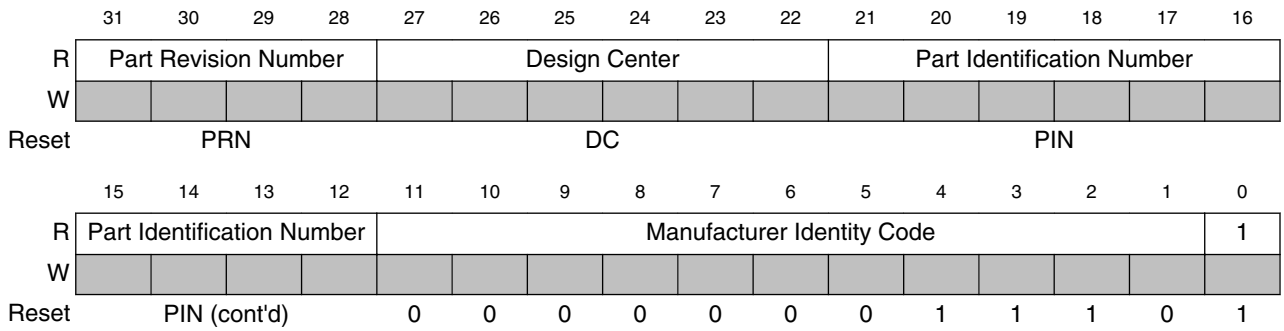


Table 68-5. DID field descriptions

Bit	Name	Description
31:28	PRN	Part Revision Number These bits contain the revision number of the part
27:22	DC	Design Center These bits indicate the device design center
21:12	PIN	Part Identification Number These bits contain the part number of the device
11:1	MIC	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for NXP, 0xE.
0	Bit [0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register

68.4.1.4 Port Configuration Register (PCR)

Register index: 127

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FPM	Reserved	MCKO_EN	MCKO_DIV			EVT_EN	Reserved	Reserved	0	0	0	ALC_NUM_LN		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	0	0	0	0	0	Reserved	Reserved	0	0	0	0	0	0	0	PSTAT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCR, shown in the figure above, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 68-7. PCR field descriptions

Bit	Name	Description
31	FPM	Full Port Mode The value of the FPM bit determines if Nexus Aurora mode is enabled. 0 Nexus Aurora mode disabled 1 Nexus Aurora mode enabled
30	Reserved	
29	MCKO_EN	The value of the MCKO_EN bit determines if Nexus Aurora mode is enabled. 0 Nexus Aurora mode disabled 1 Nexus Aurora mode enabled
28:26	MCKO_DIV	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. Set to 000 when Nexus Aurora mode is enabled.
25	EVT_EN	EVTO/EVTI Enable This bit enables the EVTO/EVTI port functions. 0 EVTO/EVTI port disabled 1 EVTO/EVTI port enabled

Table continues on the next page...

Table 68-7. PCR field descriptions (continued)

Bit	Name	Description
24	Reserved	
23	Reserved	
19:18	ALC_NUM_LN	Aurora Lane Control, Number of Lanes The value of this bit determines how many high-speed Aurora lanes are enabled 00 0 Lanes Enabled 01 2 Lanes Enabled 10 4 Lanes Enabled 11 Reserved
15	Reserved	
9	Reserved	
8	Reserved	
0	PSTAT_EN	Processor Status Mode Enable[1] This bit enables processor status (PSTAT) mode. In PSTAT mode, all auxiliary output port MDO pins are used to transmit processor status information, and Nexus messaging is unavailable. 0 PSTAT disabled 1 PSTAT mode enabled

1. PSTAT Mode is intended for factory processor debug only. The PSTAT_EN bit should be written to disable PSTAT mode if Nexus messaging is desired. No Nexus messages are transmitted under any circumstances when PSTAT mode is enabled.

68.4.1.5 Aurora Configuration Register (ACR)

Register index: 126

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LVDS_RXOPT[3:0]				0	0	0	0	LVDS_TXOPT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The ACR, shown in the figure above, is used to configure the RXOPT and TXOPT settings of the LVDS receive and transmit pads.

Table 68-9. ACR Field Descriptions

Bit	Name	Description
15:12	LVDS_RXOPT[3:0]	Configures LVDS receive pad options rxopt3, rxopt2, rxopt1, rxopt0
7:0	LVDS_TXOPT[7:0]	Configures LVDS transmit pad options txopt7, txopt6, txopt5, txopt4, txopt3, txopt2, txopt1, txopt0

68.5 Functional Description

68.5.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the output port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation.

For SoCs that support message transmission via the Nexus Aurora Link, asserting the MCKO_EN and FPM bits and setting the NUM_ALC_LN bits to a non-zero value enables NAL data transmission.

The following table describes the NPC reset configuration options.

Table 68-10. NPC reset configuration options

Power On Reset status asserted?	JCOMP asserted high	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register[1]	NUM_ALC_LN bits of the PCR > 0	Configuration
yes	X	X	X	X	Reset
no	no	X	X	X	Reset
no	yes	0	X	X	Disabled
no	yes	1	0	X	Reduced Port Mode

1. This bit is not available

68.5.2 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 test access port (TAP) for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in Figure 68-3 . The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in Figure 68-4.

The instructions implemented by the NPC TAP controller are listed in the following table. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 68-11. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

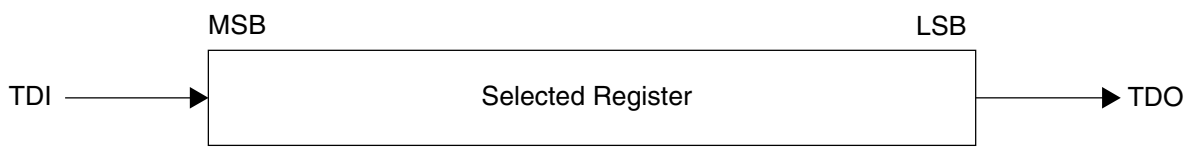
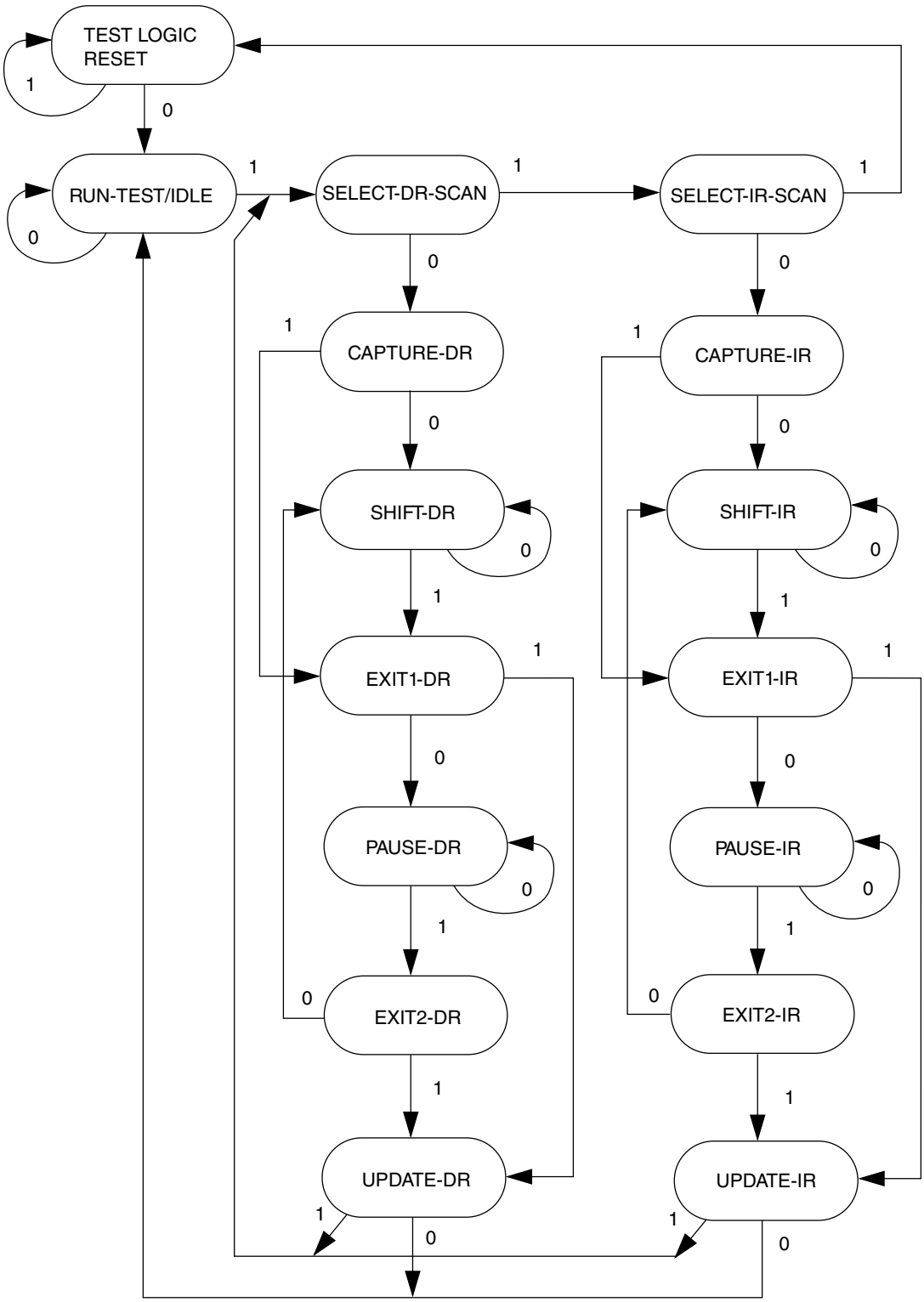


Figure 68-2. Shifting Data Into Register

68.5.2.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or setting JCOMP to a value other than the NPC enable encoding resets the NPC TAP controller. When not in power-on reset, the NPC TAP controller is enabled by driving JCOMP with the NPC enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 68-3. IEEE 1149.1-2001 TAP Controller State Machine

68.5.2.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the output port or shifted out serially by accessing the Nexus Device ID register through the TAP.

Transmission of the device identification message serially via the TAP is achieved by performing a read of the register contents as described in Selecting a Nexus Client Register. The device identification message is not transmitted on the Aurora Link in Aurora Link mode, and it can be retrieved only via the TAP in that mode.

68.5.2.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 68-4, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Table 68-12 illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

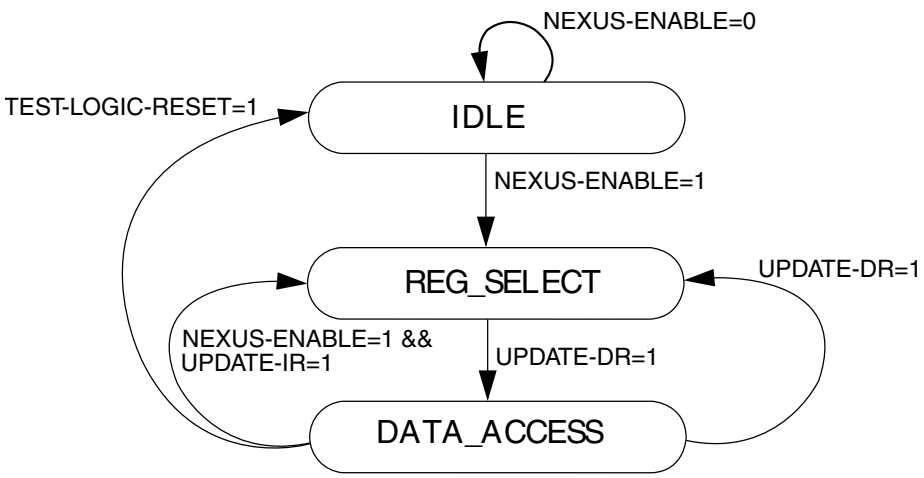


Figure 68-4. NEXUS Controller State Machine

Table 68-12. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction

Table continues on the next page...

Table 68-12. Loading NEXUS-ENABLE instruction (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
8	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
9	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
10	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

68.5.2.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following table. The read/write control bit is set to 1 for writes and 0 for reads.

MSB	LSB
7-bit register index	R/W

Figure 68-5. IEEE 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the JTAG shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 68-13. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

68.5.3 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

68.5.4 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

68.5.5 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. If JCOMP is negated, an internal reset is asserted, indicating that all Nexus modules should be held in reset.

68.5.6 Data Transmission via Nexus Aurora Link

The NPC supports data transmission over a high-speed serial Nexus Aurora Link (NAL). The NPC's role in Aurora Link data transmission is to route trace data to the NAL while continuing to manage arbitration between Nexus clients for the right to transmit data. In addition, the NPC monitors the NAL transmit queue and will halt further data transmission from Nexus clients if the queue is in danger of an overrun condition. In the event of an overrun condition, the NPC transmits a standard Nexus error message.

Once the PCR is configured for Aurora Link Mode (MCKO_EN bit set, FPM bit set, ALC_NUM_LN set to non-zero value), the NPC waits for the NAL to acknowledge it is ready to receive data before starting data transmission. All data transmitted in Aurora Link mode is transmitted in frames, with each frame consisting of 8192 bytes of data. The first beat of each frame is accompanied by the assertion of the start of frame control signal, and the last beat of each frame is accompanied by the assertion of the end of frame control signal. Each beat of valid NAL data from the NPC is also accompanied by the assertion of the NAL data valid message signal.

Data received by the NAL is held in a FIFO until it can be sent out on the Aurora Link. The NPC keeps track of the NAL FIFO fill level. To help prevent an overrun of the NAL FIFO, the NPC holds off further grants to all clients if the NAL FIFO is filled above the watermark level. In the event that the NAL FIFO is overrun with data, the NPC transmits its FIFO overrun error message.

68.6 Initialization/Application Information

68.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2001 standard for more detail.

Chapter 69
Nexus Aurora Link (NAL)

69.1 Chip-specific NAL information

On this chip, the reset value of NAL General Status Register (NAL_GSR) is 0x0000_2400h.

69.2 Introduction

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. The following figure details the NAL block diagram. The number of lanes, n, is 4.

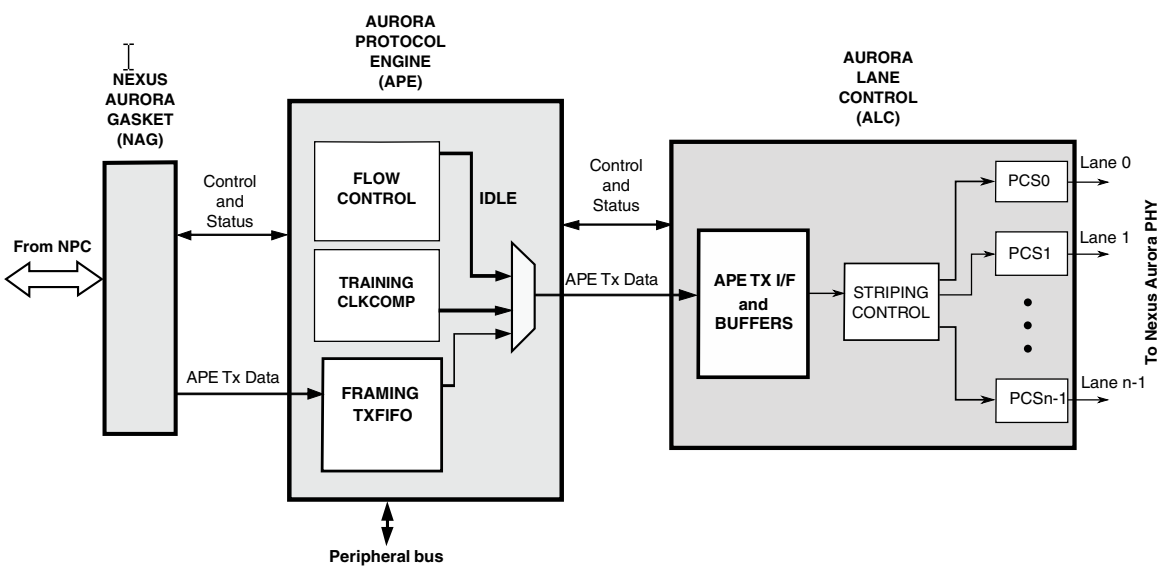


Figure 69-1. NAL block diagram

Debug data in the Nexus format from multiple clients is collected by the Nexus Port Controller (NPC) ; messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NPC/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. The table below lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 69-1. NAL and NAP feature summary

Sub-Component	Feature
NPC Aurora Gasket (NAG)	• Transmit FIFO to buffer data as it crosses the NPC/NAL clock-domain boundary and Transmit FIFO space available indication to NPC. The NPC uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. • Platform to Domain clock domain crossing. The NPC is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	• Link training • Per the Aurora Protocol specification, clock-compensation, and data framing
Aurora Link Control (ALC)	• Striping of data stream from the Aurora core across all available lanes in two-byte chunks • PCS functionality for NAL
Nexus Aurora PHY (NAP)	• Physical transmission of serial data to device pins • External clock source for SerDes clock domain

69.3 Transmit operation

During transmit operation, the NPC collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NPC places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol

Each payload is transferred from the NPC to the NPC/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each end symbol packet transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

NOTE

Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

69.4 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

69.5 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note

Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Write 80000000h to NAL_GCR
2. Write 00000000h to NAL_GCR
3. Write 80783C0Fh to NAL_TCR
4. Write 80000000h to NAL_GCR

5. Write 00000000h to NAL_GCR
6. Write A0080000h to NPC_PCR

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage. Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

69.6 Clock compensation considerations

The Aurora Protocol Specification describes clock compensation as a means for the link to compensate for clock rate differences between the transmitter and receiver. The specification requires that a clock compensation be inserted at a minimum every 10,000 symbol pairs. This allows the link to accommodate up to a 200 ppm difference in clock rates. The GCR[CCOEN] field sets the default clock compensation time out at 7424 cycles. However, due to the fact that the NAL is restricted to only inserting clock compensation sequences after an end-of-frame, it may not be possible to always meet the Aurora specification requirement.

Since the NPC always operates in framed mode, there is one scenario that can prevent an adequate number of clock compensation sequences from being generated: the sparse data case. In the sparse data case, there are long periods of time when no data is flowing through the link; no data means no frames. The NAL could end up transmitting idles for an undetermined number of cycles while waiting for the NPC to provide new data. However, a user can get around this problem by insuring that the NPC always has at least some minimal amount of data to send. In extreme sparse data cases where clock compensation is required, the receiver could reset the NPC or use a periodic watchpoint to force data transmission, and therefore force and end-of-frame to complete clock compensation.

69.7 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 69-2. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0h	NAL General Status Register (NAL_GSR)	32	R	See section
8h	NAL General Control Register (NAL_GCR)	32	R/W	0000_0000h
Ch	NAL Training Control Register (NAL_TCR)	32	R/W	0000_0000h

69.7.1 Nexus Aurora Link General Status Register (NAL_GSR)

The NAL_GSR is a read only register that describes the status of the NAL. See the chip-specific NAL information for the NAL_GSR reset value.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												Reserved	0		
W																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			TXCFG			0									AS
W																

Table 69-3. NAL_GSR field descriptions

Field	Description
31–20 Reserved	Reserved This bitfield is reserved.
19 Reserved	Reserved This bitfield is reserved.
18–13 Reserved	Reserved This bitfield is reserved.
12–10 TXCFG	TX Lane Configuration. Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved

Table continues on the next page...

Table 69-3. NAL_GSR field descriptions (continued)

Field	Description
	011 4 TX lanes 100 Reserved 101 Reserved 110 Reserved 111 Reserved
9–1 Reserved	Reserved This bitfield is reserved.
0 AS	Aurora Status. 0 Aurora is not enabled. 1 Aurora is enabled.

69.7.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RST	ALCPD	0												FCM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PCRST	0	SLD			0	BOOE	BOOD				0	CCOEN	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 69-4. NAL_GCR field descriptions

Field	Description
31 RST	Reset the Aurora Channel. 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
30 ALCPD	ALC/NAL Power Down. Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect.

Table continues on the next page...

Table 69-4. NAL_GCR field descriptions (continued)

Field	Description
	1 NAL is powered down and inactive.
29–18 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.
17 FCM	Flow Control Mode. 0 Completion Mode 1 Immediate Mode
16-15 Reserved	Reserved and must be written with zero.
14 PCRST	Protocol Converter Reset. Resets the ALC when asserted.
13 Reserved	Reserved This read-only bit is reserved and always has the value zero.
12–10 SLD	Symbol Lock Delay. 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read
9 Reserved	Reserved This read-only bit is reserved and always has the value zero.
8 BOOE	Bond Offset Override Enable. 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
7–4 BOOD	Bond offset override data. Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset 0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset

Table continues on the next page...

Table 69-4. NAL_GCR field descriptions (continued)

Field	Description
	1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
3 Reserved	Reserved and must be written with zero.
2 CCOEN	Clock Compensation Override Enable. This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles
1–0 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.

69.7.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in Static training.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STE	AHD	BHD	VHD	Reserved					ATC				Reserved					BTC				Reserved						VTC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 69-5. NAL_TCR field descriptions

Field	Description
31 STE	Static Training Enable. 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.
30 AHD	Hold in Align. 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
29 BHD	Hold in Bond. 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared. (applies even during duplex-mode training)
28 VHD	Hold in Verify. 0 Follow Aurora training sequence

Table continues on the next page...

Table 69-5. NAL_TCR field descriptions (continued)

Field	Description
	1 Remain in verify until this bit is cleared. (applies even during duplex-mode training)
27–23 Reserved	Reserved This bitfield is reserved and must be written with zero.
22–19 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles 0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
18–14 Reserved	Reserved This bitfield is reserved.
13–10 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles 0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles

Table continues on the next page...

Table 69-5. NAL_TCR field descriptions (continued)

Field	Description
	1111 Remain in bond for 3072 cycles
9–4 Reserved	Reserved This bitfield is reserved.
3–0 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles 0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

Chapter 70
Nexus Aurora PHY (NAP)

70.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Port Controller (NPC) Port Configuration Register (NPC_PCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note

The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

The following figure shows a high level block diagram of the High Speed Nexus Interface and the NAP.

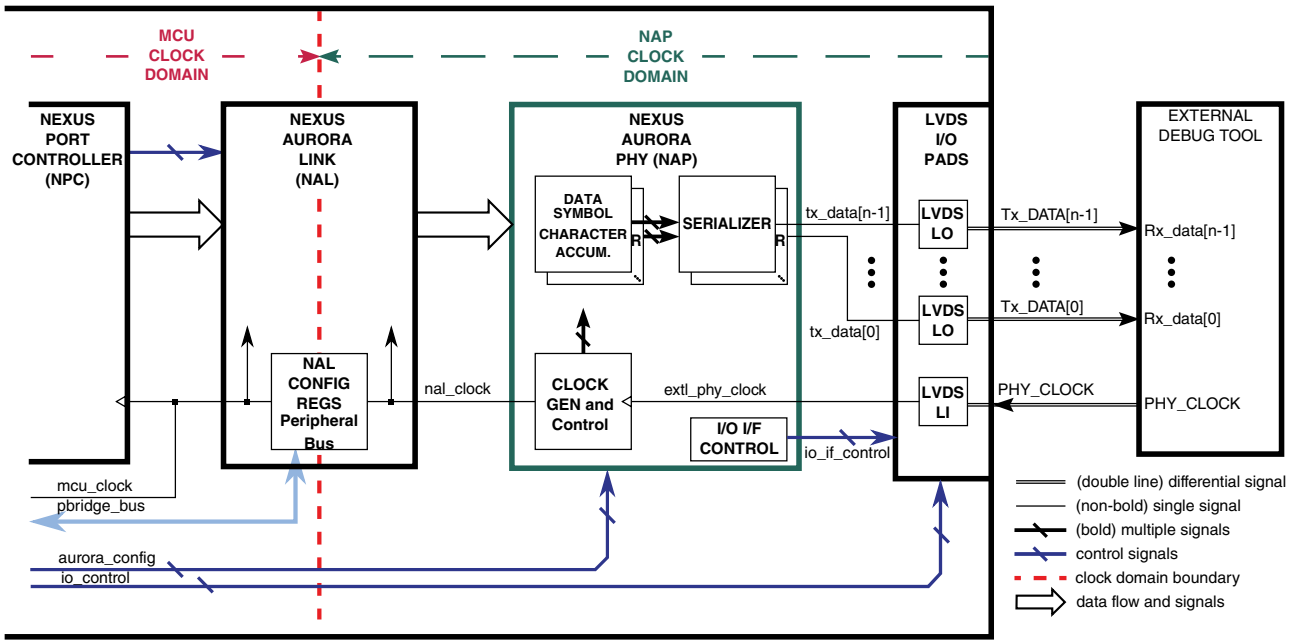


Figure 70-1. High speed Nexus interface block diagram

70.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

70.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NPC_PCR setting; functional operation is enabled by the NPC.

70.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

70.1.2.2 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

70.1.2.3 Transmit

During Nexus trace output operation, it is the responsibility of the NPC to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

70.2 External signal description

The external signals of the NAP are listed in the following table. Each of the NAP signals consists of a positive and negative differential pair.

Table 70-1. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool

Table continues on the next page...

Table 70-1. NAP signals (continued)

Signal Name	Direction	Description
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1
TX Data2 (TX2P/TX2N)	Output	NAP Tx data for lane 2
TX Data3 (TX3P/TX3N)	Output	NAP Tx data for lane 3

70.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NPC_PCR which is described in the NPC Chapter.

70.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

70.4.1 Data symbol character accumulator

The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

70.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

The following figure shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

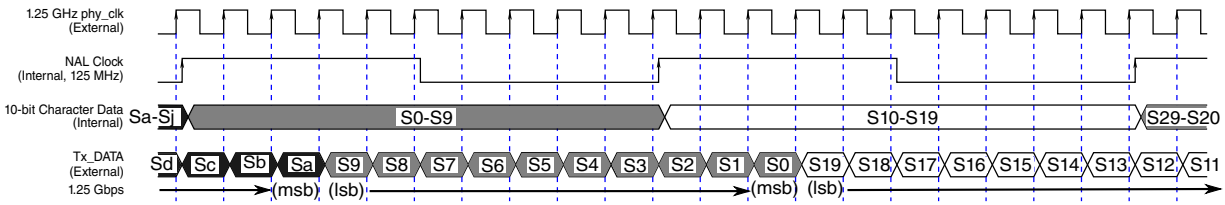


Figure 70-2. Serializer timing diagram

70.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

70.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NPC_PCR to configure the number of lanes and LVDS pads.

Chapter 71
JTAG Controller (JTAGC)

71.1 Chip-specific JTAGC information

71.1.1 JTAGC ACCESS_AUX block instructions

This table shows the ACCESS_AUX JTAGC instructions.

Table 71-1. ACCESS_AUX 5-bit JTAG instructions

Instruction	Code[4:0]	Instruction Summary
ACCESS_AUX_x	10000–11110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NPC	10000	Enables access to the NPC TAP controller
ACCESS_AUX_z7a	10001	Enables access to Core z7a ONCE
ACCESS_AUX_z7b	10010	Enables access to Core z7b OnCE
ACCESS_AUX_ALL_CORES	10011	Enables DAISY Chained access to the Cores
Reserved	10100	Reserved
Reserved	10101	Reserved
Reserved	10110	Reserved
ACCESS_AUX_NXMC0	10111	Enables access to the Nexus XBAR Slave 0 TAP controller
ACCESS_AUX_NXMC1	11000	Enables access to the Nexus XBAR Slave 1 TAP controller
ACCESS_AUX_NXMC2	11001	Enables access to the Nexus XBAR Slave 2 TAP controller
ACCESS_AUX_OnCE	11010	Enables access to Core z4 OnCE.
Reserved	11011	Reserved
Reserved	11100	Reserved
Reserved	11101	Reserved
ACCESS_AUX_NAL	11110	Enables access to NAL

71.1.1.1 JTAG ID reset value

JTAG ID reset value for the device is 0x298D801D

71.1.2 Control 1 register

The Control 1 register is a 32-bit shift register path from TDI to TDO selected when the ENABLE_CONTROL1 instruction is active. The default reset value of the Control 1 register is 0. The Control 1 register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	NXMC1_SEL	PSTAT_SEL		TE2	TE1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TE0	TEZ7B	TEZ7A	TEZ4	Reserved						TMZ7B		TMZ7A		TMZ4	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The following table describes the Control 1 register functions.

Table 71-2. Device identification register field descriptions

Field	Description
NXMC1_SEL	Select bit for NXMC1 Trace Source 0 Flexray is traced via NXMC1 1 Ethernet is traced via NXMC1
PSTAT_SEL	Processor Status Select. Determines which core's Processor status is reflected in SSCM's debug status registers 6 and 7 and also in NPC's pstat messages. 01 Core Z7a's status is reflected 10 Core Z7b's status is reflected Others Core Z4's status is reflected
TE2	Nexus XBAR Slave 2 Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TE1	Nexus XBAR Slave 1 Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TE0	Nexus XBAR Slave 0 Timestamp Enable. 0 Timestamping is disabled

Table continues on the next page...

Table 71-2. Device identification register field descriptions (continued)

Field	Description
	1 Timestamping is enabled
TEZ7B	Z7b core Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TEZ7A	Z7a core Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TEZ4	Z4 core Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
Reserved	This field is reserved.
TMZ7B	Timestamp Mode for Z7b core. Defines Timestamping granularity. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 64th message
TMZ7A	Timestamp Mode for Z7a core. Defines Timestamping granularity. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 64th message
TMZ4	Timestamp Mode for Z4 core. Defines Timestamping granularity. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 64th message

71.2 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

71.2.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to Register description for more information about the JTAGC registers.

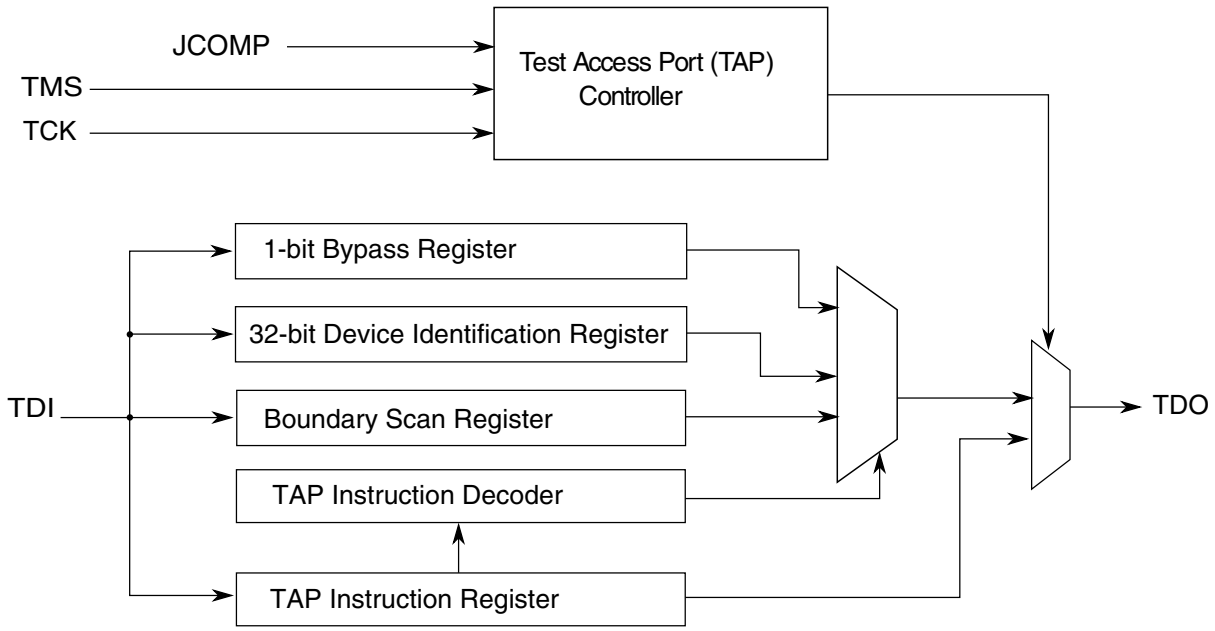


Figure 71-1. JTAG (IEEE 1149.1) block diagram

71.2.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 71-6 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Control 1 register

- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

71.2.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

71.2.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

71.2.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

71.2.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

71.3 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 71-3. JTAG signal properties

Name	I/O	Function	Reset State
TCK	Input	Test Clock	Weak pullup
TDI	Input	Test Data In	Weak pullup
TDO	Output	Test Data Out	High Z[1]
TMS	Input	Test Mode Select	Weak pullup
JCOMP	Input	JTAG Compliancy	Weak pulldown

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

71.3.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

71.3.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

71.3.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

71.3.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

71.3.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

71.4 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

71.4.1 Instruction register

The JTAGC block uses a 5-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in

the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 00001b , making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	4	3	2	1	0
R	0	0	0	0	1
W	Instruction Code				
Reset:	0	0	0	0	1

Figure 71-2. Instruction register

71.4.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

71.4.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
Reset	PRN				DC						PIN					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											1
W																
Reset	PIN (contd.)				MIC											1

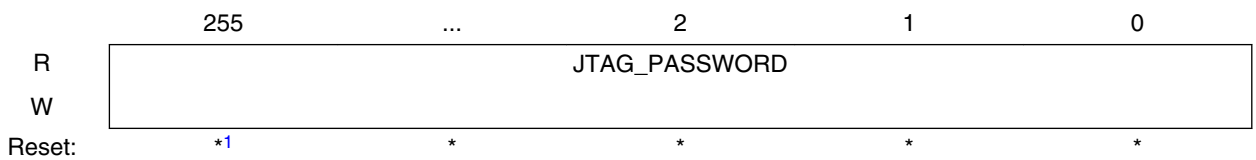
The following table describes the device identification register functions.

Table 71-4. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x26.
PIN	Part Identification Number. Contains the part number of the device. 0011011000b.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

71.4.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 71-5. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

71.4.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for

bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

71.5 Functional description

This section explains the JTAGC functional description.

71.5.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

71.5.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

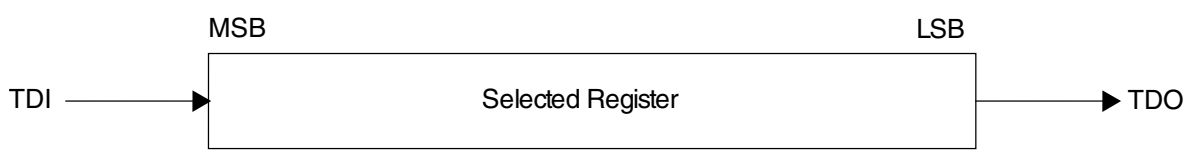
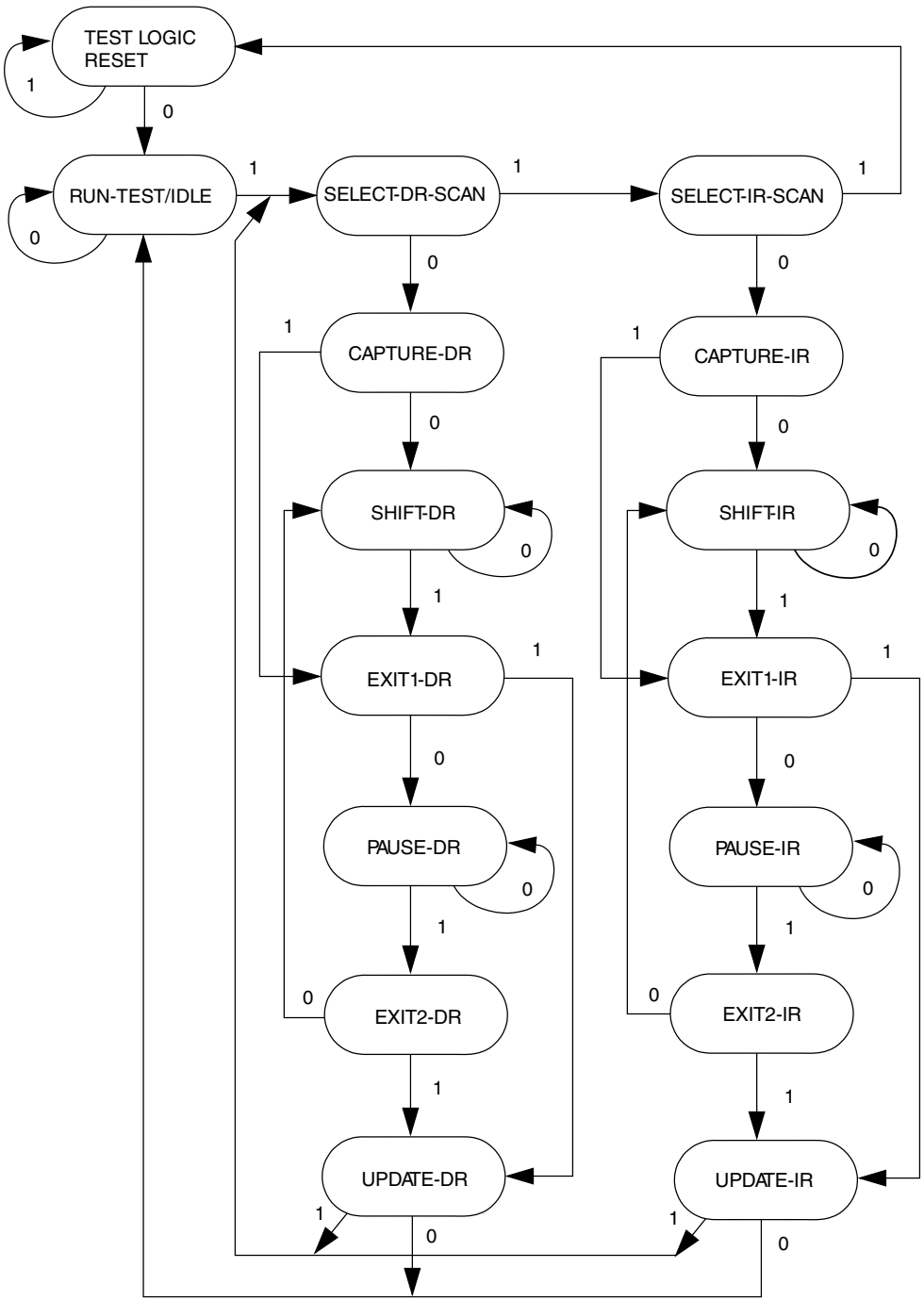


Figure 71-3. Shifting data through a register

71.5.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the

TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 71-4. IEEE 1149.1-2001 TAP controller finite state machine

71.5.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

71.5.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

71.5.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 71-6. General 5-bit JTAG instructions

Instruction	Code[4:0]	Instruction Summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	00101	Intended for factory debug only
Factory debug reserved	00110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	00111	Selects JTAG_PASSWORD register
HIGHZ	01001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	01010	Intended for factory debug only

Table continues on the next page...

Table 71-6. General 5-bit JTAG instructions (continued)

Instruction	Code[4:0]	Instruction Summary
CLAMP	01100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
ENABLE_CONTROL1	01110	Selects Control 1 register
Factory debug reserved	01111	Intended for factory debug only
BYPASS	11111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

71.5.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

71.5.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

71.5.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

71.5.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

71.5.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

71.5.4.6 ENABLE_CONTROL1 instruction

The ENABLE_CONTROL1 instruction selects the Control 1 register for connection as the shift path between TDI and TDO.

71.5.4.7 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

71.5.4.8 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

71.5.4.9 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

71.5.4.10 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

71.5.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

71.6 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

Chapter 72
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

72.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

72.2 Abbreviations

Table 72-1. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State

Table continues on the next page...

Table 72-1. Abbreviations (continued)

Abbreviation	Expansion
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

72.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as "Compact JTAG" or "CJTAG".

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths

- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

72.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- **Advanced**
  - Two-pin operation using the "Advanced Protocol" — signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

72.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors..
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

72.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in the following table. Signal names that end in “C” are associated with expanded functionality (beyond the functionality provided by the IEEE 1149.1 standard).

The table uses the following notation:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the following pins:

- TCKC
- TDIC
- TDOC
- TMSC

Table 72-2. TAP.7 signal list

Signal name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
Mandatory signal count (minimum)		2	4	4	4	4

72.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in the following figure. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP. 7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

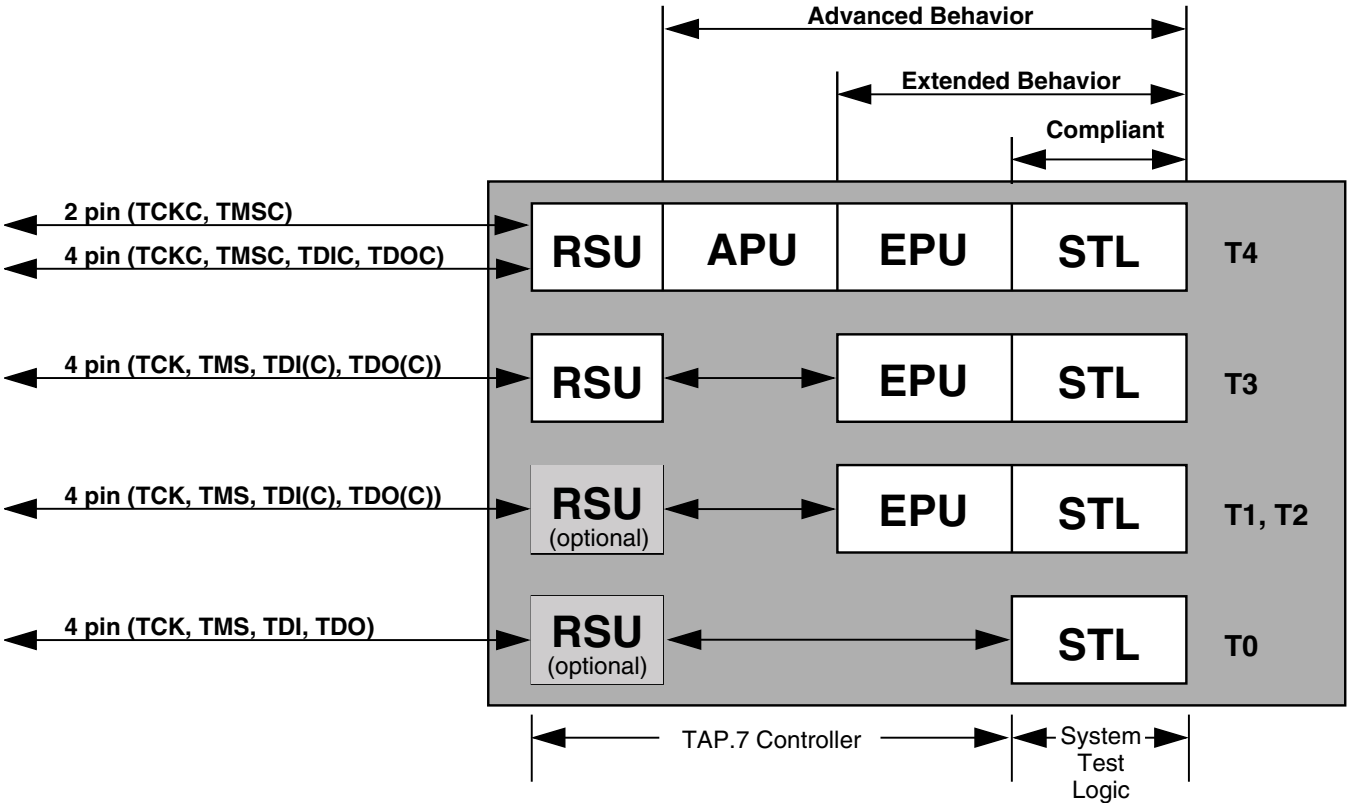


Figure 72-1. TAP.7 Controller Architecture

72.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in the following figure.

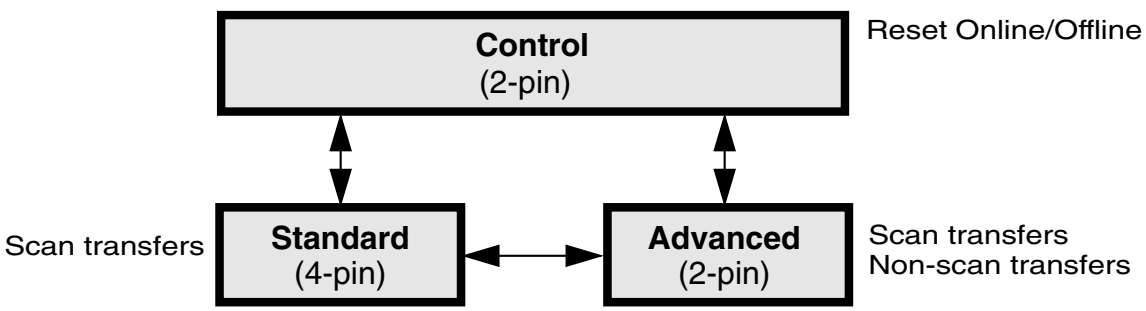


Figure 72-2. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and

vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in the following table.

Table 72-3. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

72.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

72.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

72.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

72.4 Operating models

The following figure illustrates a simplified view of the CJTAG operating model.

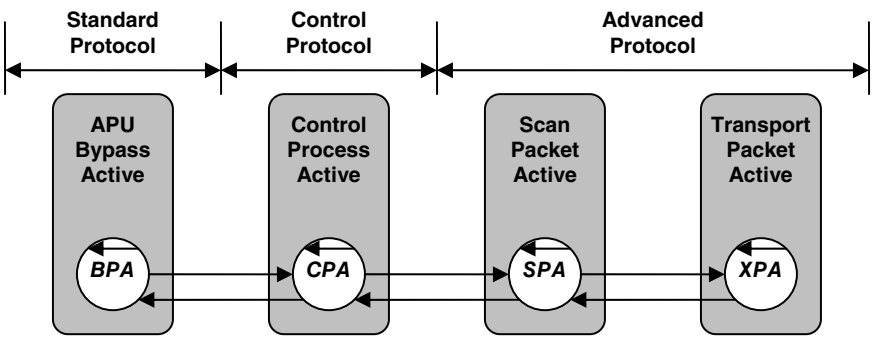


Figure 72-3. CJTAG operating model

72.5 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

72.5.1 T0 functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

72.5.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

72.5.3 T2 functions

- "Super Bypass" function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

72.5.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

72.5.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

72.6 Ancillary services

72.6.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options

- Escape sequences
- TAPC state machine

72.6.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC.

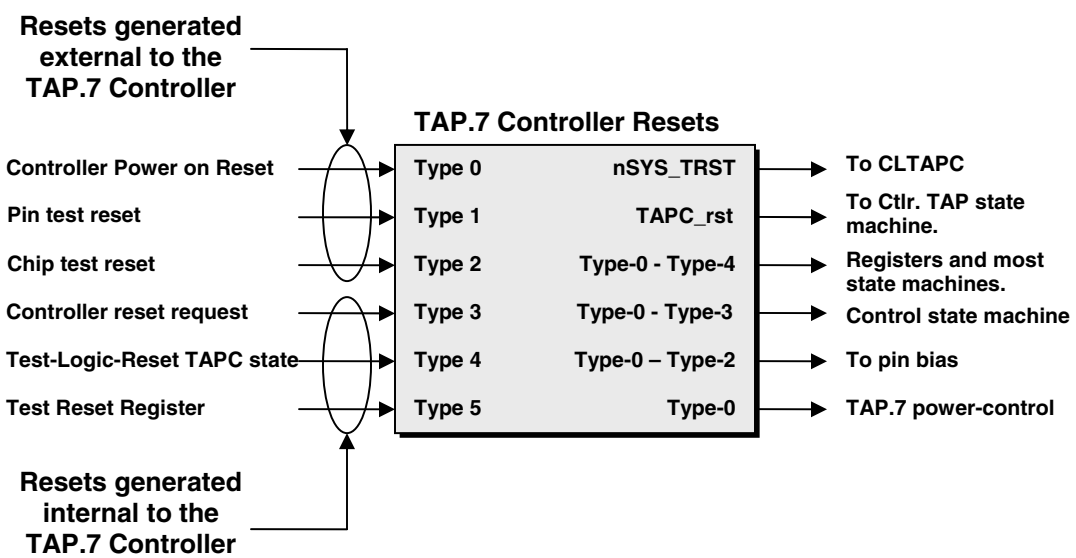


Figure 72-4. Conceptual view of the EPU reset logic and state machines

72.6.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

72.6.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

72.6.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

72.6.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

72.6.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

72.6.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

72.6.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

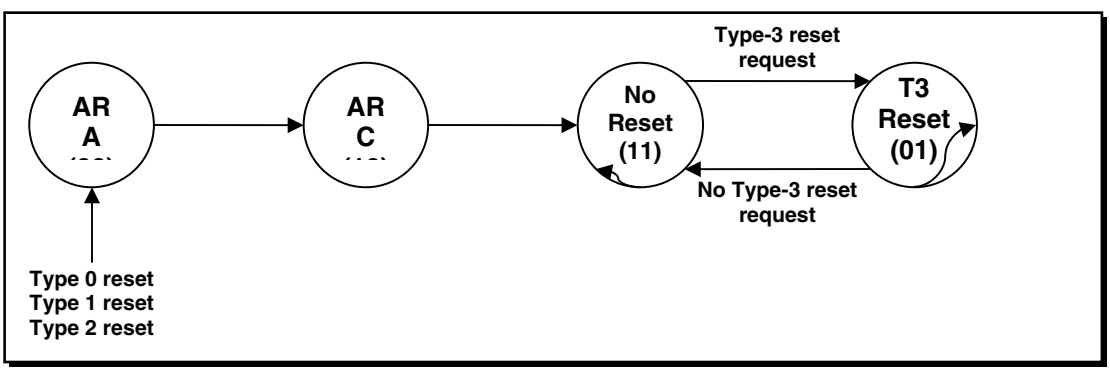


Figure 72-5. Reset state machine

Table 72-4. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in the following table.

Table 72-5. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

72.6.3 Start-up Options

72.6.3.1 Overview

The TAP.7 start-up options are shown in the following table. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 72-6. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled

Table continues on the next page...

Table 72-6. Start-up Options (continued)

Start-up Option	Description
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

72.6.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 72-7. Reset values for TAP.7 startup options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

72.6.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

72.6.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP. 7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in the following figure.

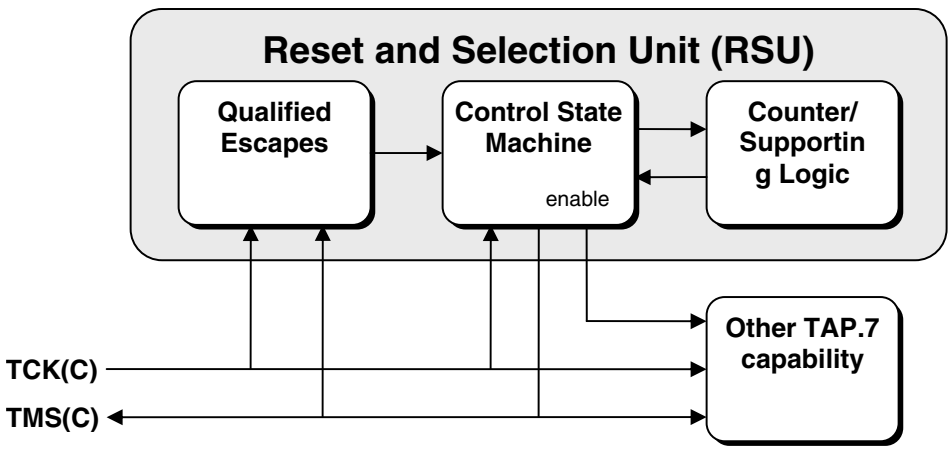


Figure 72-6. Conceptual view of technology selection mechanism

72.6.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

72.6.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

72.6.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

72.6.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

72.6.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

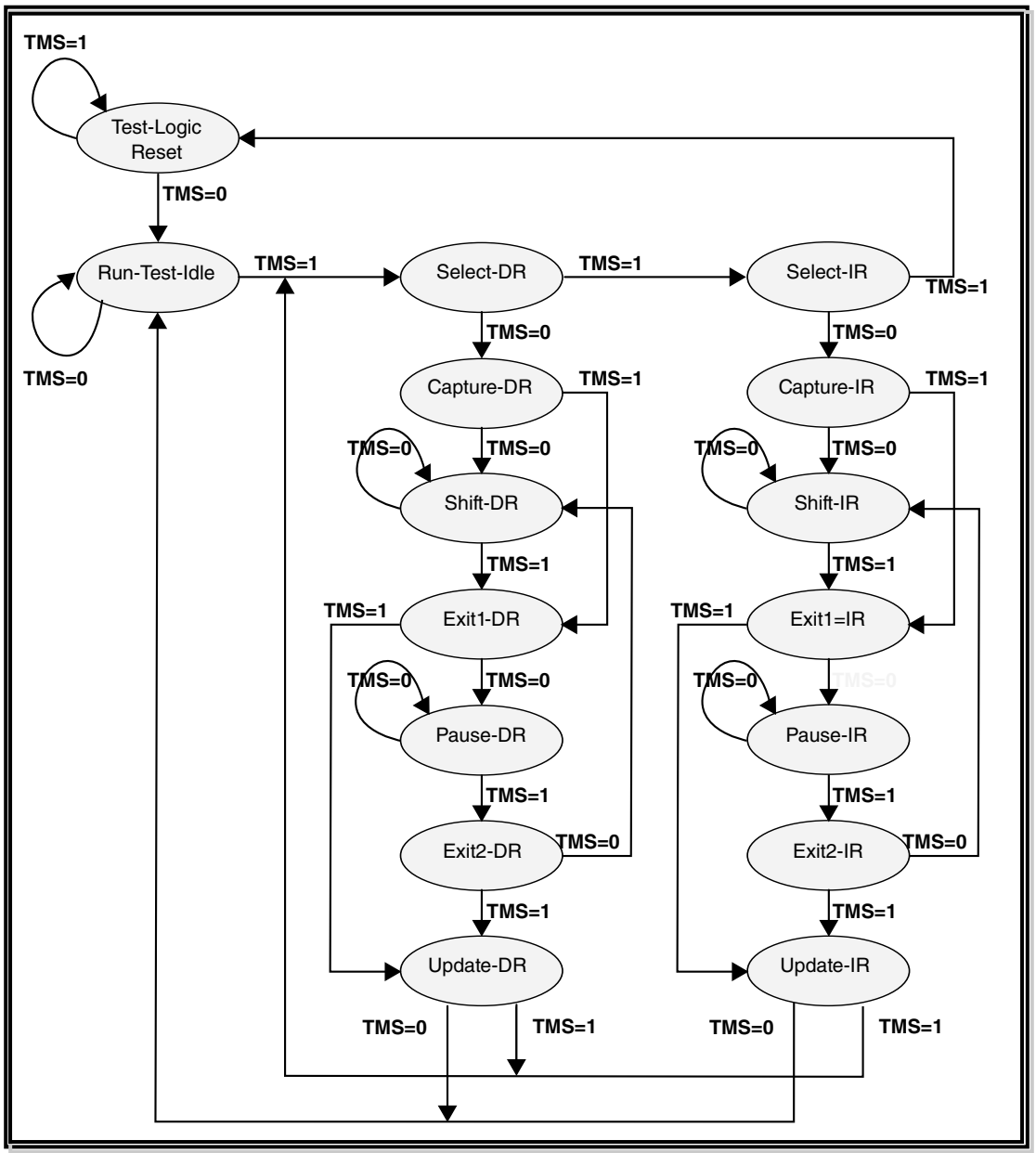


Figure 72-7. Falling edge TAP controller state machine

72.7 EPU (Extended Protocol Unit) Operation

72.7.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

72.7.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

72.7.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

72.7.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

72.7.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths

6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control levels.

72.7.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

72.7.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

72.7.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP. 7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

72.7.2 EPU Registers

72.7.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

72.7.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 72-8. TAP.7 controller register list (managed with commands)

Register type	Global/local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	RDBACK1	Read-back 1
		32	CNFG0	Configuration Register 0

M = Mandatory register

O = Optional register

72.7.2.3 Register Descriptions

72.7.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

72.7.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

1 – Suspend use of control levels. Enables "System Path."

0 – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

72.7.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

1 – Inhibit ZBS detection.

0 – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

72.7.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 72-9. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	**JScan0:** • 1149.1 Compliant Operation. • Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	**JScan1:** • 1149.1 Compatible Operation • Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	**JScan2:** • 1149.1 Compatible Operation. • Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	**JScan3:**

Table continues on the next page...

Table 72-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
		• Non-compatible 1149.1 Operation • Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. • Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. • Enables the use of SSDs
T4	01000	**OScan0** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01001	**OScan1** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01010	**OScan2** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01011	**OScan3** • TAP.1 components • No TDI in non-shift states • No TDO in any state
T4	01100	**OScan4** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01101	**OScan5** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01110	**OScan6** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications

Table continues on the next page...

Table 72-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
	01111	**OScan7** • TAP.1 components • No TDI in non-shift states • No TDO in any state
	10000	**MScan** • Data rate dependent components • Multi TAP.7 communication • Directed CID assignment • Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	**Reserved**

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

72.7.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

0 – Scan selection directives are not enabled.

1 – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

72.7.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the "Global Register."

72.7.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

72.7.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

0 – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.

1 – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

72.7.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

0x – TDI and TDO have JTAG function.

1x – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

72.7.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

72.7.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

72.7.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

72.7.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as "0".

Table 72-10. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

72.7.2.3.14 Configuration Register 0 (CNFG0)

Table 72-11. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4

Table continues on the next page...

Table 72-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision.
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes

Table continues on the next page...

Table 72-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported

Table continues on the next page...

Table 72-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
Reserved	31:29	3	Reserved	Reserved, read as zero

72.7.2.3.15 Register reset values

Table 72-12. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DCA	DCA	DCA	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

72.7.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

72.7.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

72.7.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

72.7.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of Shift-DR states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in the following figure.

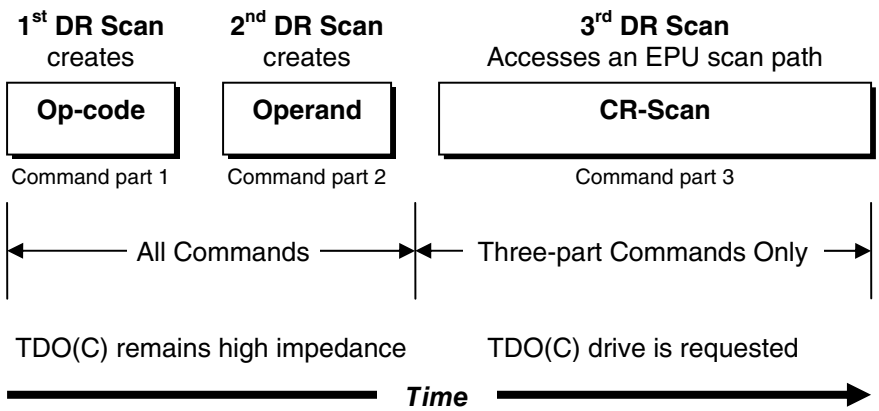


Figure 72-8. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

72.7.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in the following table.

Table 72-13. – TAP.7 controller command list

Command class	Opcode count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate[1]	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

72.7.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing an external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

72.7.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

72.7.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

72.7.3.5 Command Definitions

72.7.3.5.1 Store Commands

72.7.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 72-14. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y

Table continues on the next page...

Table 72-14. Store Miscellaneous Control (continued)

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

72.7.3.5.1.2 Store Conditional 1-bit (STC1)

Table 72-15. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	**Operand**	**c**	**bbb**	**CAPE**	**Description**
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	–-	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

72.7.3.5.1.3 Store Conditional 2-bit (STC2)

Table 72-16. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	**Operand**	**c**	**bb**	**CAPE**	**Description**
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	–-	0	No change
			00	1	Reserved
			01	1	STCKDC = vv

Table continues on the next page...

Table 72-16. Store Conditional 2 bit (STC2) (continued)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
			10	1	APFC = vv
			11	1	Reserved

72.7.3.5.1.4 Store Scan Format (STFMT)

Table 72-17. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

72.7.3.5.2 Select Commands

This section describes the select commands with their operands.

72.7.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 72-18. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

72.7.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 72-19. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

72.7.3.5.3 Scan Commands

72.7.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 72-20. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

72.7.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 72-21. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only

Table continues on the next page...

Table 72-21. Scan String (SCNS) (continued)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

72.7.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in the following figure:

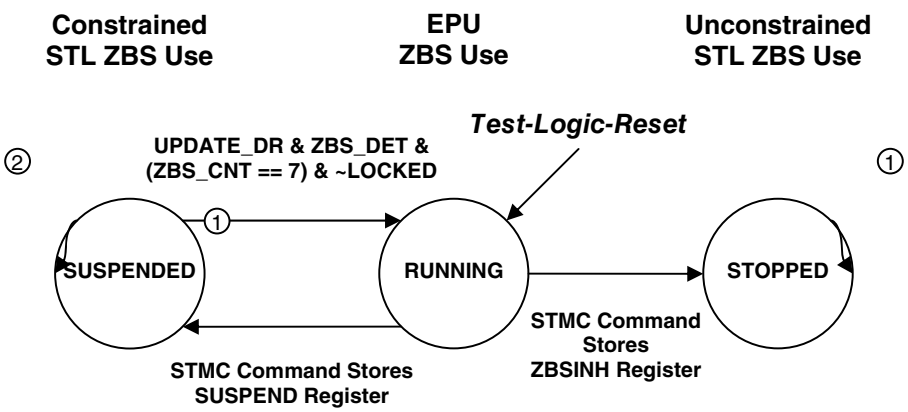


Figure 72-9. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in the following table. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 72-22. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL

Table continues on the next page...

Table 72-22. ZBS use state characteristics with both TAP.7 and other ZBS use (continued)

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

72.7.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

72.8 APU (Advanced Protocol Unit) Operation

72.8.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Two groups of scan formats (MScan and OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan Scan Format is broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a

special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

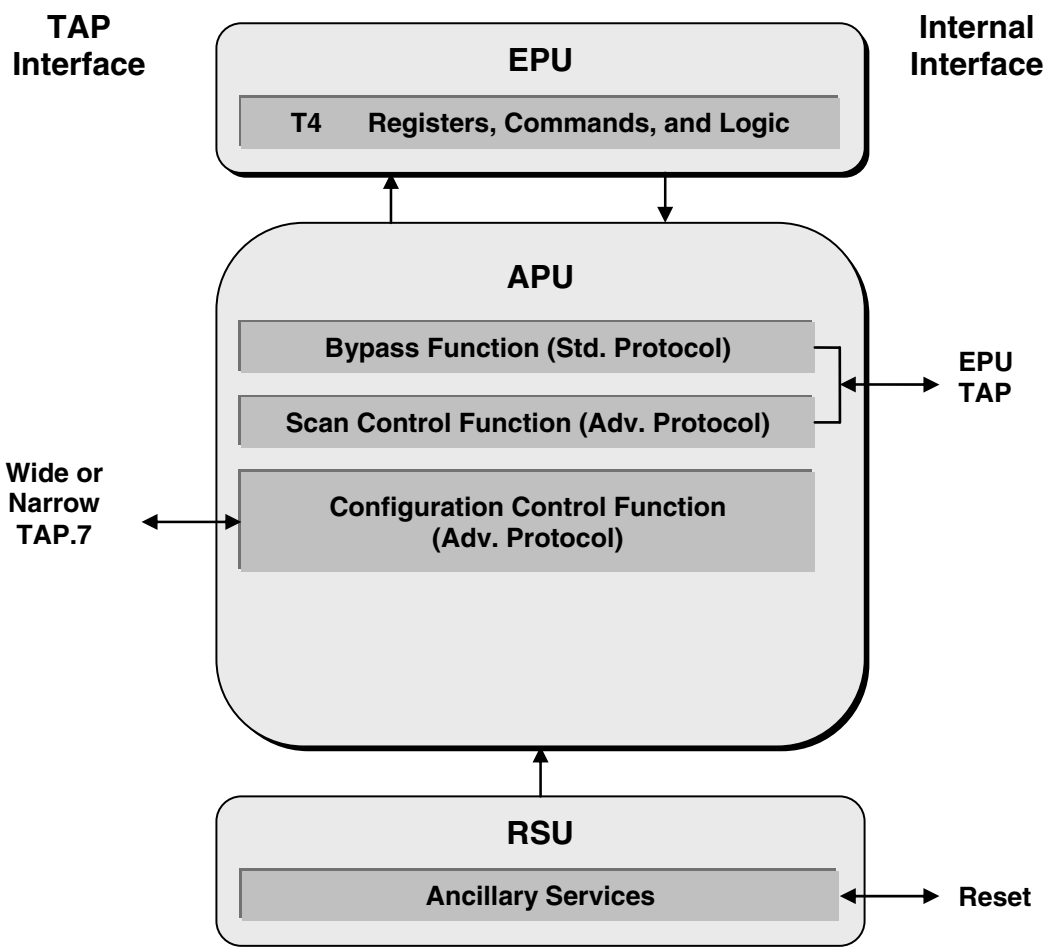


Figure 72-10. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

72.8.2 Operation

The conceptual view of the APU operation is shown in the following figure.

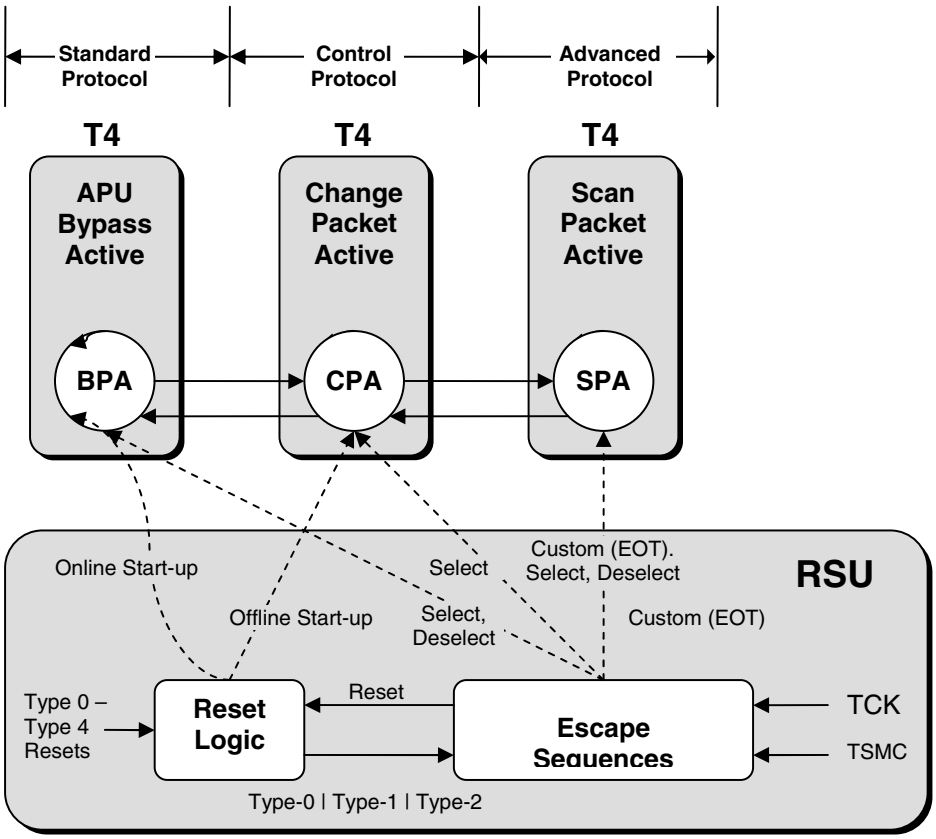


Figure 72-11. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

72.8.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

72.8.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

72.8.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

72.8.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

72.8.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

72.8.5 APU Functions

72.8.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7 or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

72.8.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

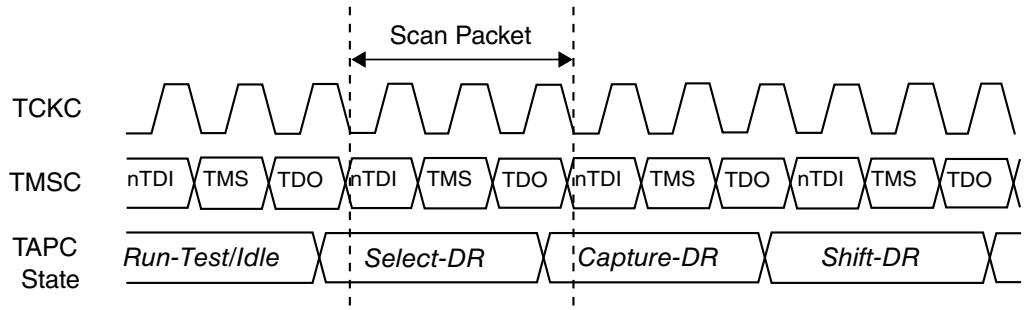


Figure 72-12. Scan Packet Sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

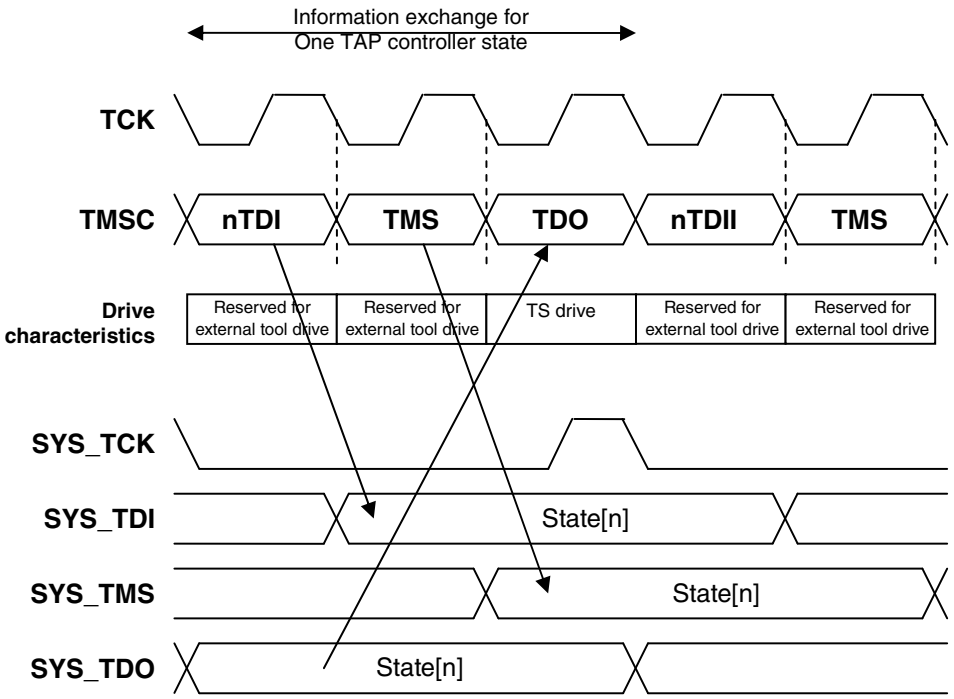


Figure 72-13. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug

applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

72.8.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

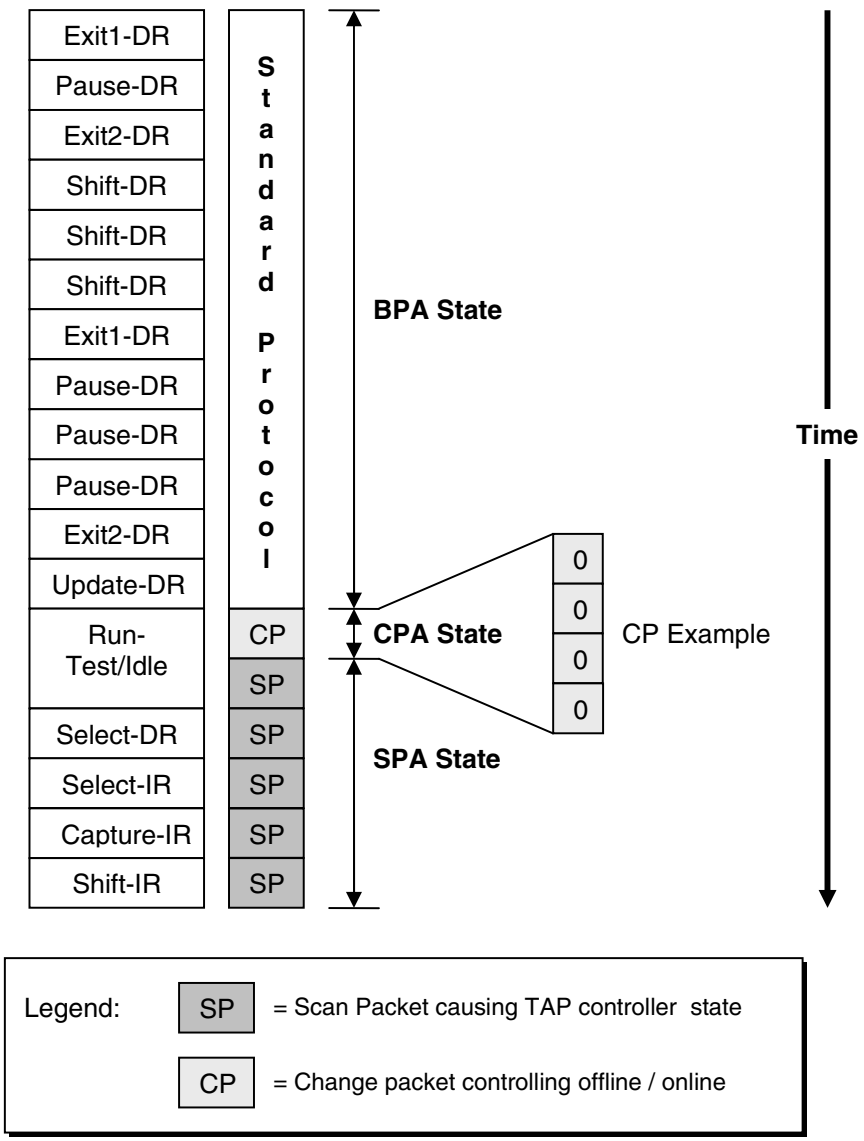


Figure 72-14. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

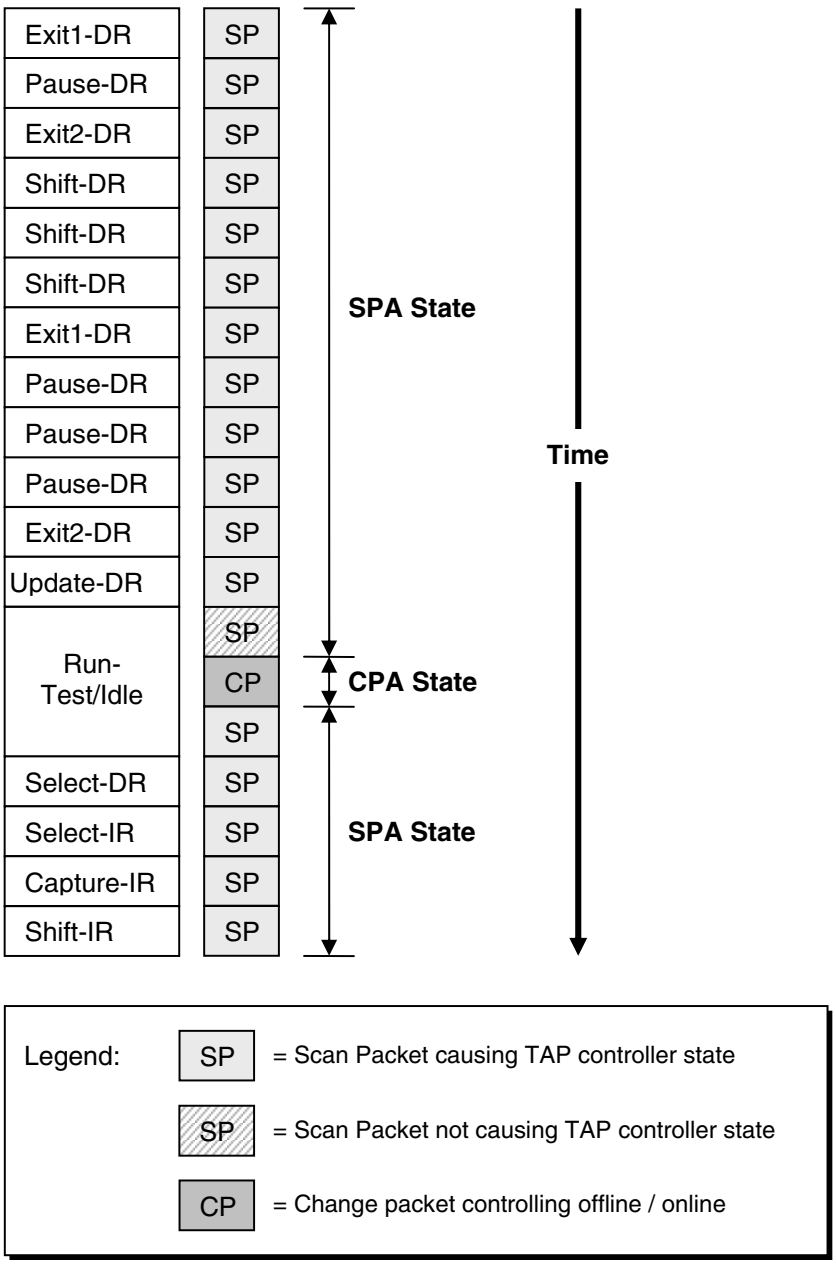


Figure 72-15. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while

the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

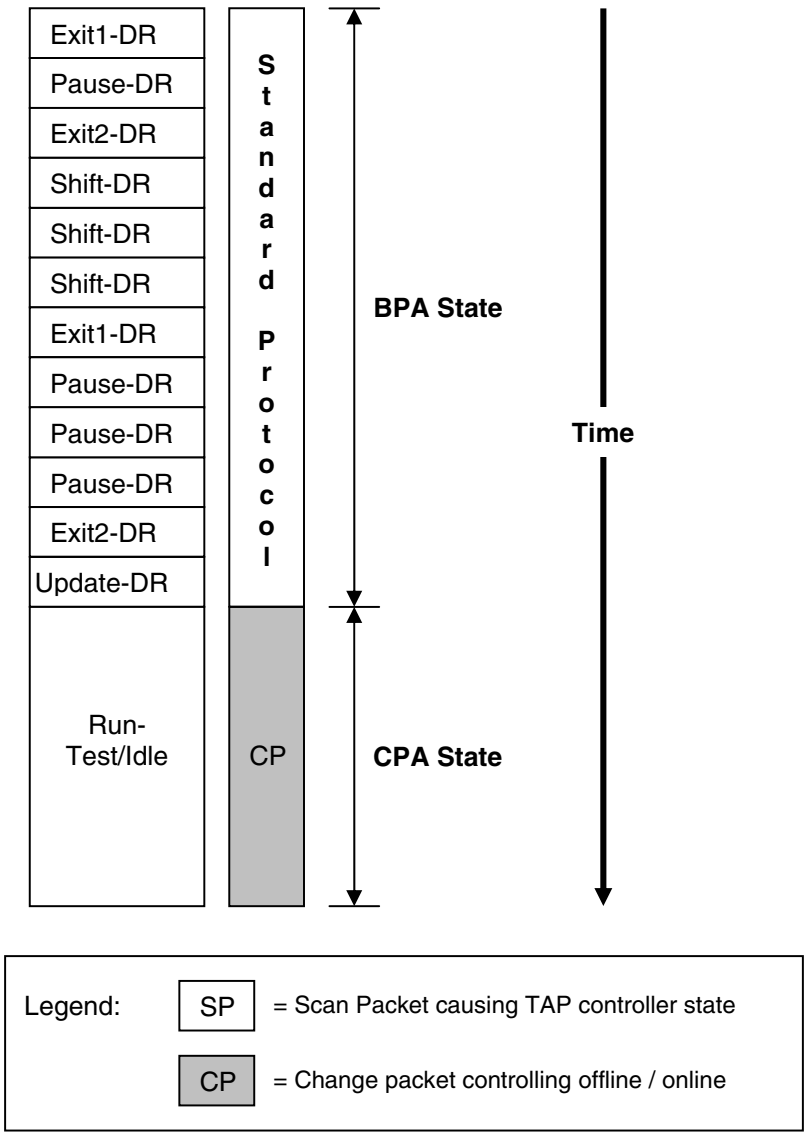


Figure 72-16. CPA State Entry from the BPA State with Offline Operation

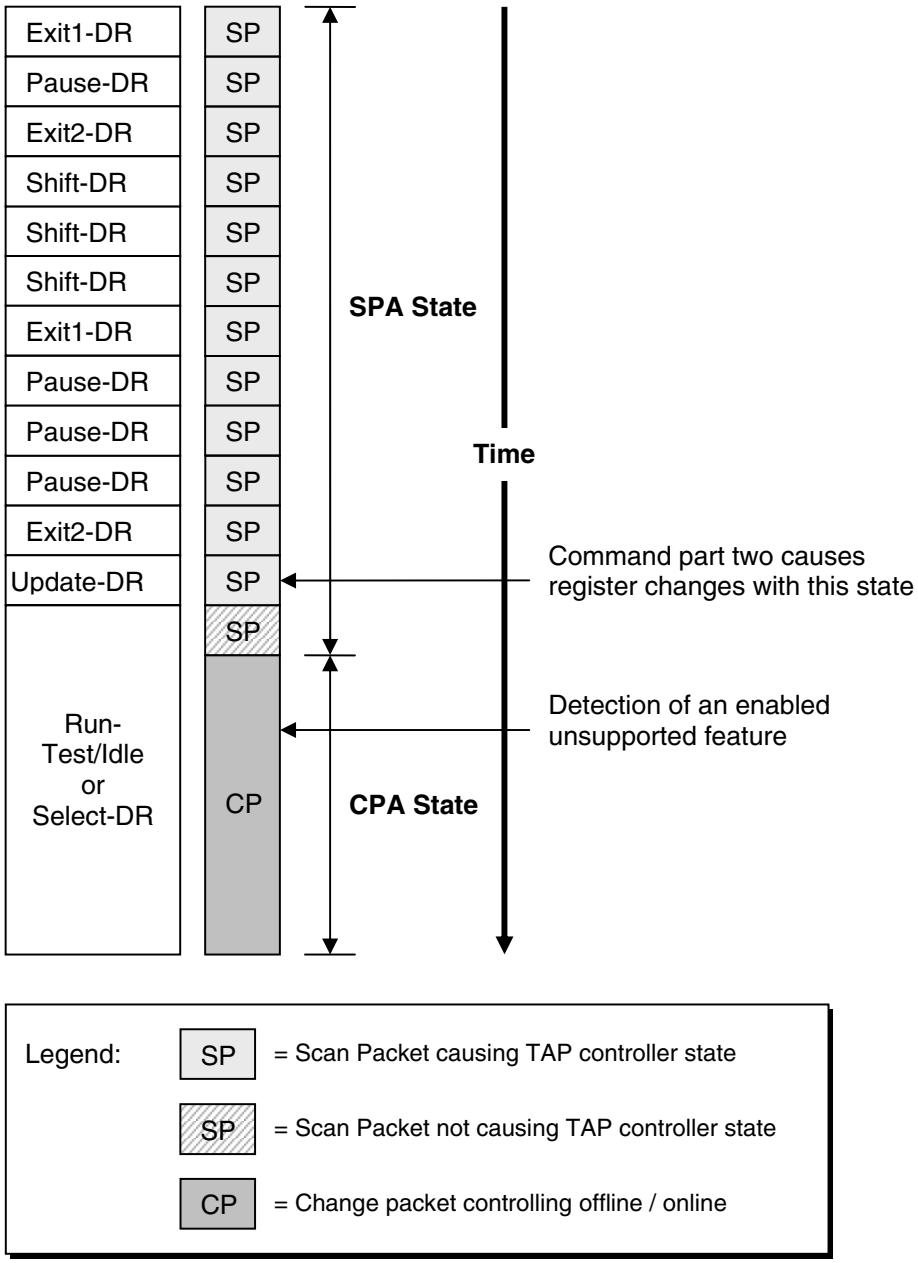


Figure 72-17. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

72.8.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.

- **Body** – Two or more bits representing one or more CP directives
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit.
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

72.8.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

72.8.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

72.8.8.1 Payload Element Content

72.8.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain and input bit frame, an output bit frame, or both. When an input bet frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

72.8.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in the following table.

Table 72-23. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

72.8.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

72.8.8.3 Scan Formats

72.8.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is

required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in Table 72-24 and Table 72-26. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

72.8.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 72-24. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			

Table continues on the next page...

Table 72-24. MScan scan format use case summary (continued)

Scan format	Supporting	Performance	Flexibility	TCKC source
	Multi-chip debug communication			
	Controller ID Allocation			

Table 72-25. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

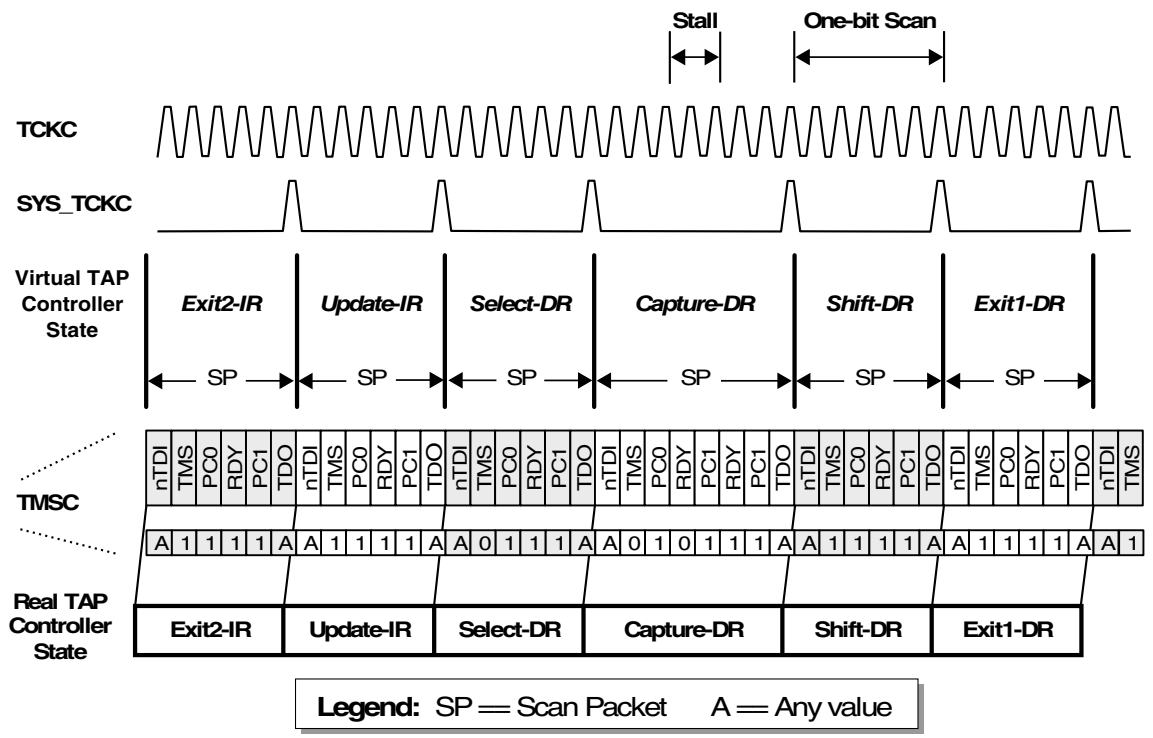


Figure 72-18. MScan Transaction

72.8.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

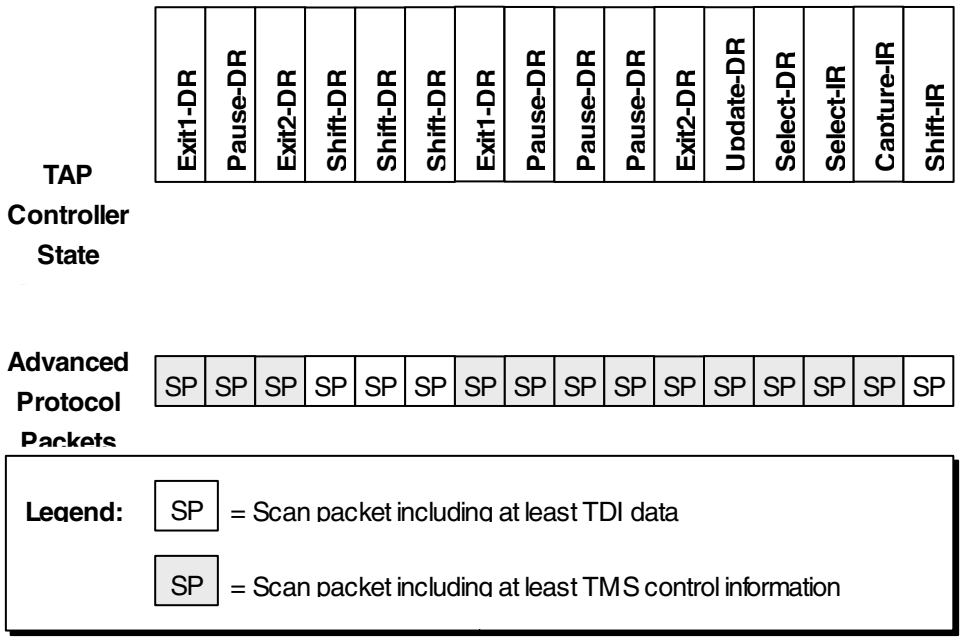


Figure 72-19. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods fo this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in the following table.

Table 72-26. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 72-27. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

72.9 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

72.9.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in Control levels.

After the control level is set tocontrol level 2, TAP.7 commands can be executed as described in EPU Commands. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in Control Process Active (CPA) Function and Configuration Change Packets (CP). Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in Store Conditional 2-bit (STC2).

Chapter 73
JTAG Master (JTAGM)

73.1 Chip-specific JTAGM information

JCOMP pin needs to be kept LOW to use JTAGM module. If JCOMP pin is driven HIGH (by debugger or otherwise), JTAGM loses control over the debug port. Either external JTAG pins or JTAGM can be active at any given point of time. Once debugger is connected, the JTAGM can not be used without power-on-reset.

73.2 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

73.2.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface . It then sends this data on TDI, TMS and TCK to the JTAGC. The JTAGM also receives serial data through TDO from the JTAGC. The JTAGM has the following registers:

- Configuration register
- Status register

- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

This figure shows the block diagram of the JTAGM.

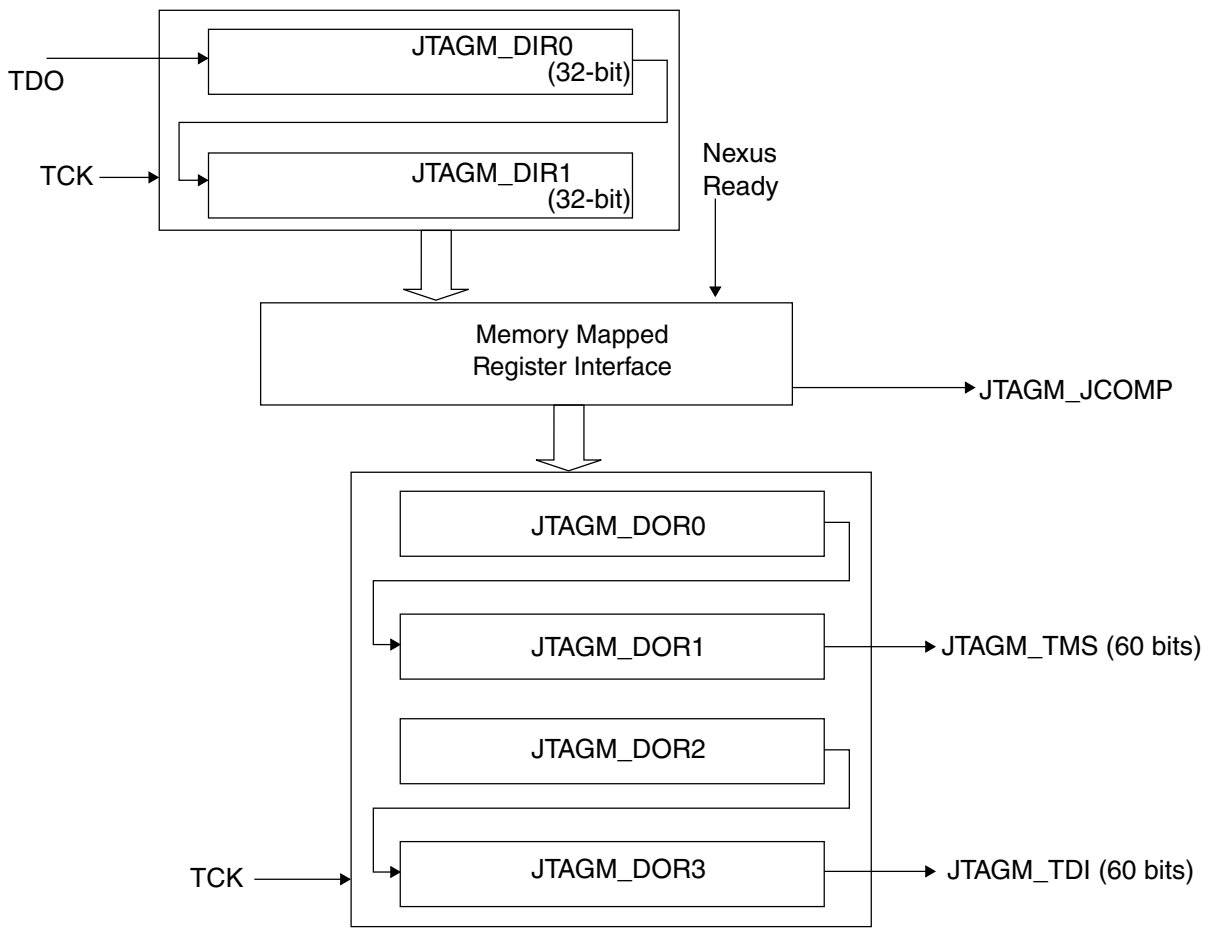


Figure 73-1. JTAGM block diagram

73.2.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG

73.3 Functional description

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

73.3.1 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available.

73.3.1.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

73.3.1.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

73.3.2 JTAGM to JTAGC serial interface

This figure shows the data transfer timings between JTAGM and JTAGC.

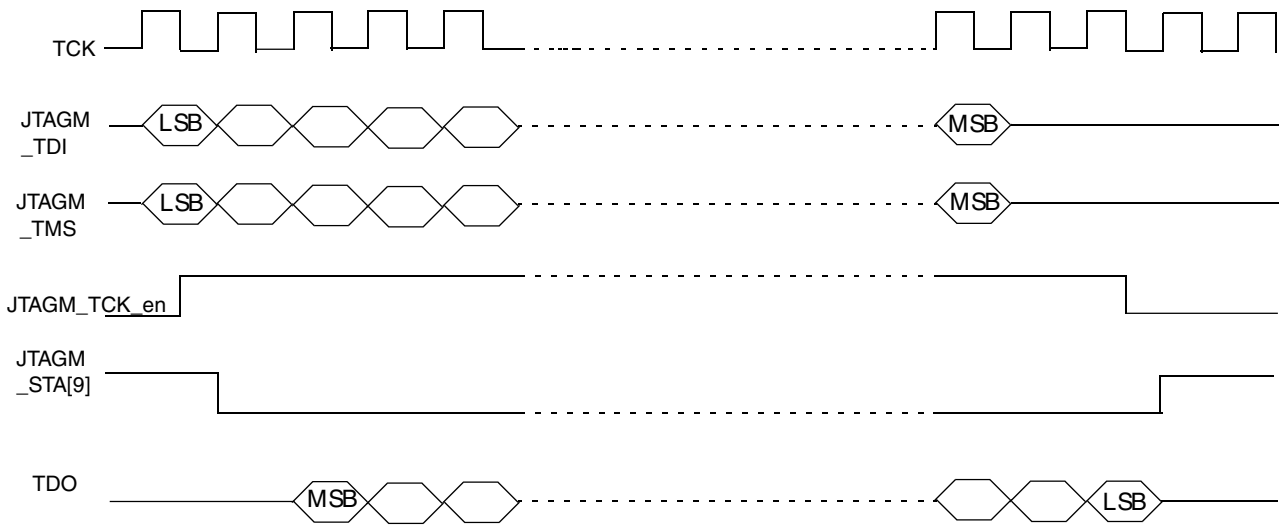


Figure 73-2. JTAGM to JTAGC serial interface diagram

73.3.3 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

73.4 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

73.5 Memory mapped registers

JTAGM supports only 32-bit write accesses to its registers. A byte write access will be converted into a 32-bit write; all other byte values in the register will become 0000_0000b.

JTAGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JTAGM_MCR)	32	R/W	0300_0002h	73.5.1/3637
4	Status Register (JTAGM_SR)	32	R/W	0020_0200h	73.5.2/3640
8	Data Out Register 0 (JTAGM_DOR0)	32	R/W	0000_0000h	73.5.3/3641
C	Data Out Register 1 (JTAGM_DOR1)	32	R/W	0000_0000h	73.5.4/3642
10	Data Out Register 2 (JTAGM_DOR2)	32	R/W	0000_0000h	73.5.5/3642
14	Data Out Register 3 (JTAGM_DOR3)	32	R/W	0000_0000h	73.5.6/3643
1C	Data Input Register 0 (JTAGM_DIR0)	32	R	0000_0000h	73.5.7/3643
20	Data Input Register 1 (JTAGM_DIR1)	32	R	0000_0000h	73.5.8/3644

73.5.1 Module Configuration Register (JTAGM_MCR)

This register contains the status bits for the current transfer.

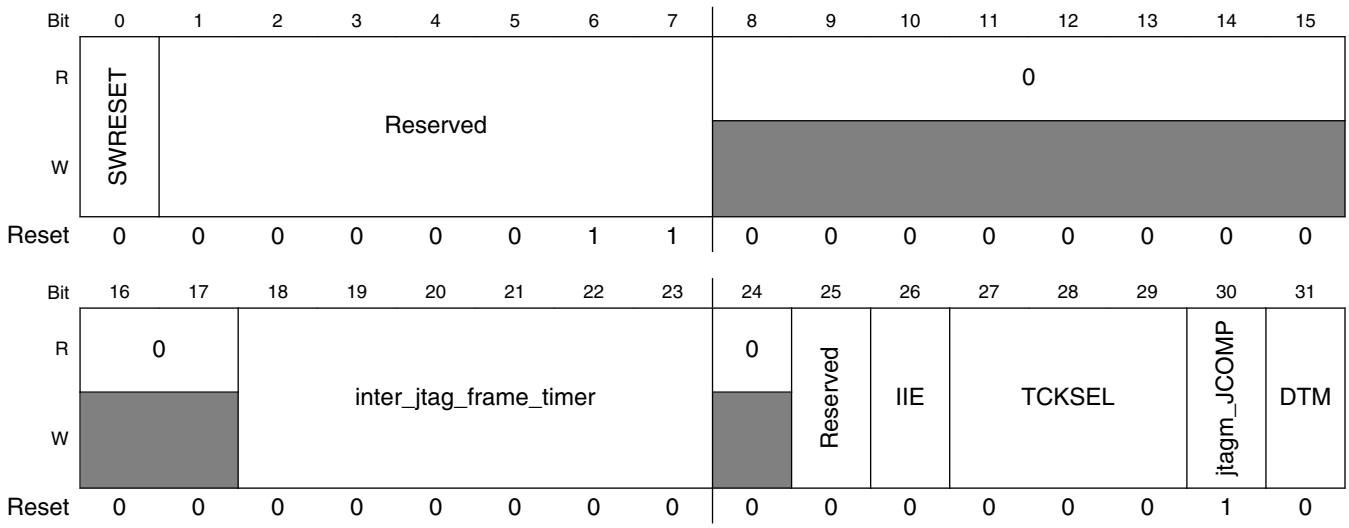
NOTE

An unexpected interrupt will occur if the Idle Interrupt is enabled after the end of frame transfer. The interrupt occurs after a JTAGM frame has completed when the JTAGM Idle Interrupt was not enabled (JTAGM_MCR[IIE] = 0b0), if the JTAGM module is then configured to enable interrupts (JTAGM_MCR[IIE] = 0b1). To clear any previous interrupts, set the Idle bit (JTAGM_SR[Idle]) before enabling a JTAGM idle interrupt (e.g. setting JTAGM_MCR[IIE] = 0b1).

NOTE

Bit-set operation on this register is prohibited as it can lead to device going to debug mode.

Address: 0h base + 0h offset = 0h



JTAGM_MCR field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit; after being written with a 1, this bit is auto-cleared within 3 TCK cycles. Writing a 0 has no effect.
1–7 Reserved	This field is reserved. The reserve bits value must not change during register write operation.
8–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 inter_jtag_frame_timer	TCK delay. 000000 0 TCK delay 000001 1 TCK delay ... ... 111111 63 TCK delay
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. The reserve bits value must not change during register write operation.
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27–29 TCKSEL	TCK clock frequency selection. When the JTAGM is enabled, this bit should not be programmed with 000. 000 Reserved 001 Reserved 010 TCK is system clock ÷ 3 011 TCK is system clock ÷ 4 100 TCK is system clock ÷ 5 101 TCK is system clock ÷ 6 110 TCK is system clock ÷ 7 111 TCK is system clock ÷ 8

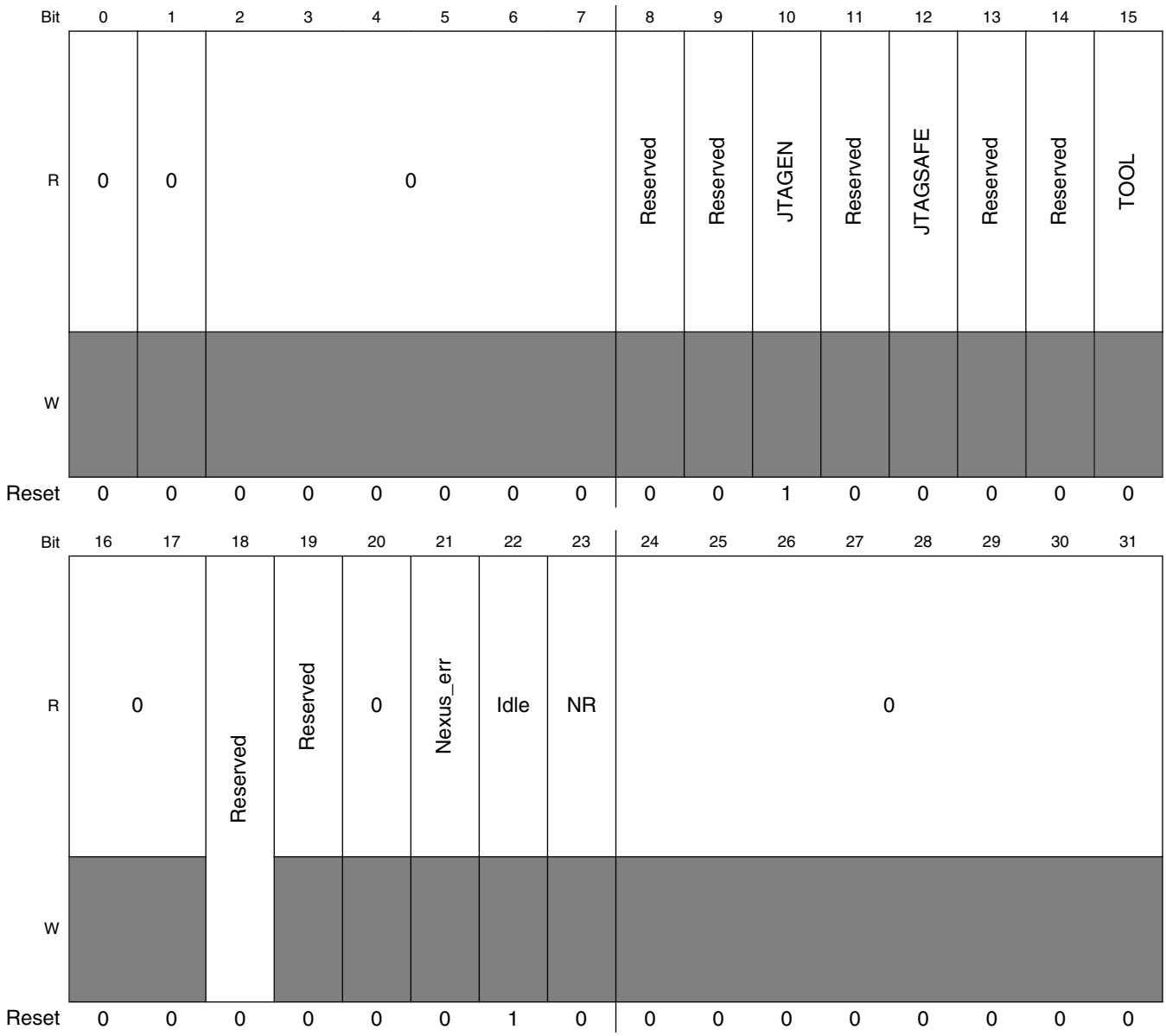
Table continues on the next page...

JTAGM_MCR field descriptions (continued)

Field	Description
30 jtagm_JCOMP	JTAG reset. 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data transferred by software disabled. 1 Data for JTAG transferred by software.

73.5.2 Status Register (JTAGM_SR)

Address: 0h base + 4h offset = 4h



JTAGM_SR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

JTAGM_SR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. Reserved to 0
9 Reserved	This field is reserved. Reserved to 0
10 JTAGEN	JTAGC status: JTAGC JTAG port enable
11 Reserved	This field is reserved. Reserved to 0
12 JTAGSAFE	JTAGC status: JTAGC JTAG safe mode
13 Reserved	This field is reserved. Reserved to 0
14 Reserved	This field is reserved. Reserved to 0
15 TOOL	JTAGC status: JTAGC tool present, set when a debug tool is connected
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 Reserved	This field is reserved. The reserve bits value must not change during register write operation.
19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.5.3 Data Out Register 0 (JTAGM_DOR0)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TMS_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR0 field descriptions

Field	Description
0–31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results.

73.5.4 Data Out Register 1 (JTAGM_DOR1)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMS_LOW																												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR1 field descriptions

Field	Description
0–27 TMS_LOW	Lower word of data for TMS, bits 27-0. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results. TMS_LOW[0] is shifted out first
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.5.5 Data Out Register 2 (JTAGM_DOR2)

Address: 0h base + 10h offset = 10h

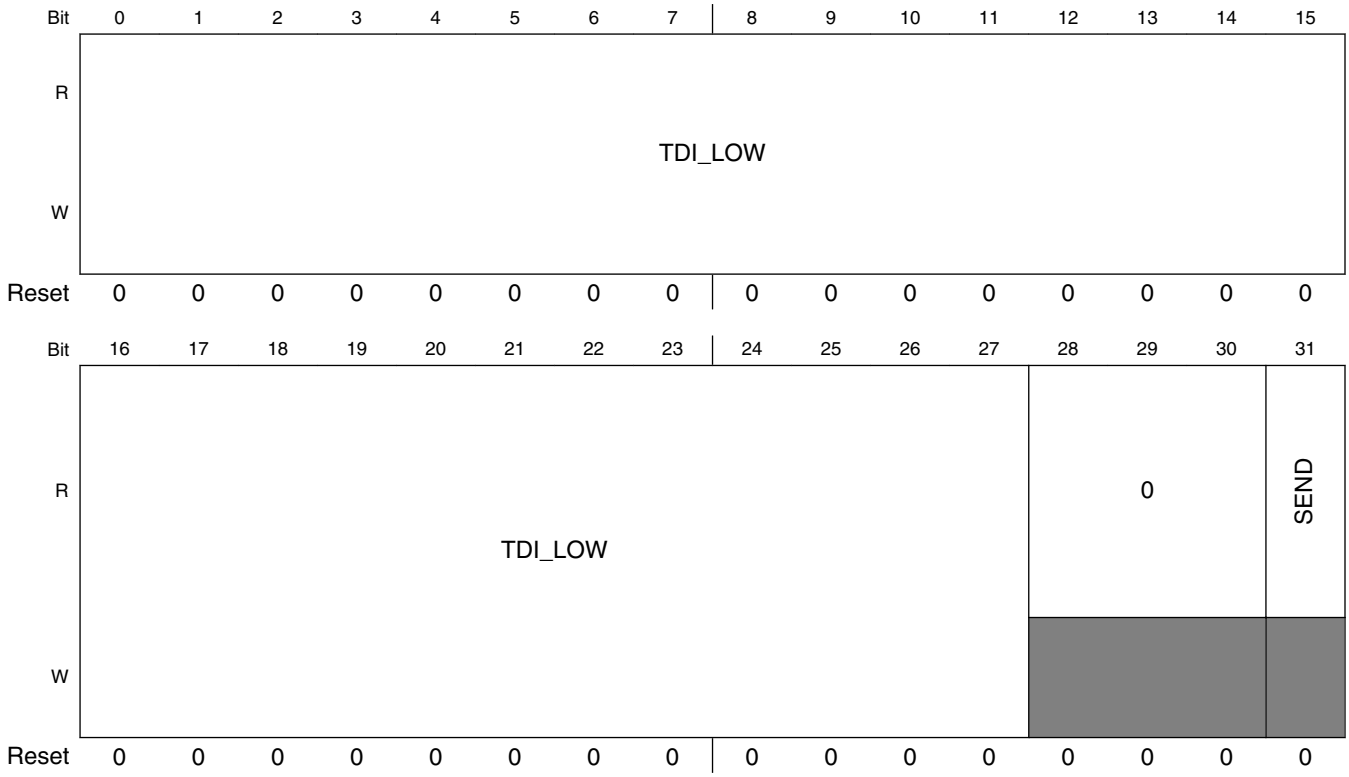
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDI_HIGH																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR2 field descriptions

Field	Description
0–31 TDI_HIGH	Higher word of data for TDI, bits 59-28. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results.

73.5.6 Data Out Register 3 (JTAGM_DOR3)

Address: 0h base + 14h offset = 14h

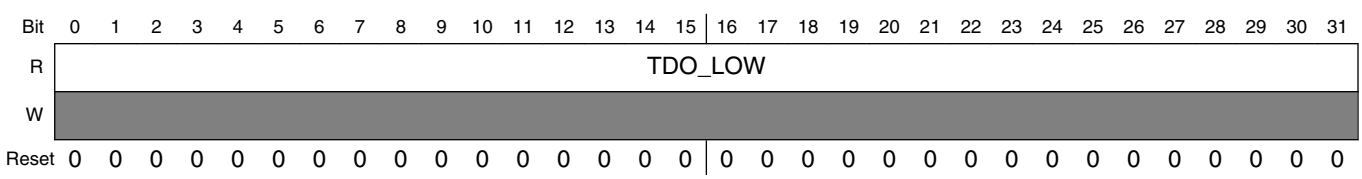


JTAGM_DOR3 field descriptions

Field	Description
0–27 TDI_LOW	Lower word of data for TDI, bits 27-0. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results. TDI_LOW[0] is shifted out first
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SEND	SEND bit When this bit is set, data is sent to the JTAGC. It is a self-clearing bit and is always read as 0. Writable by software only when DTM=1.

73.5.7 Data Input Register 0 (JTAGM_DIR0)

Address: 0h base + 1Ch offset = 1Ch



JTAGM_DIR0 field descriptions

Field	Description
0–31 TDO_LOW	Lower word of data received on TDO, bits 0-31

73.5.8 Data Input Register 1 (JTAGM_DIR1)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDO_HIGH																												Reserved			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DIR1 field descriptions

Field	Description
0–27 TDO_HIGH	Higher word of data received on TDO, bits 32-59
28–31 Reserved	This field is reserved.

Chapter 74
Cyclic Redundancy Check (CRC) Unit

74.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

74.2 Main features

- Three contexts for the concurrent CRC computation are supported
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- Four hard-wired polynomials (two CRC-8, CRC-32 ITU-T V.42, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

74.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN

- CRC-16-CCITT
- CRC-32 ITU-T V.42 (see note below)
- CRC-8-H2F Autosar polynomial

NOTE

Although the CRC-32 ITU-T V.42 polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the IEEE 802.3 standard.

74.3 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to Main features for number of contexts in this module.

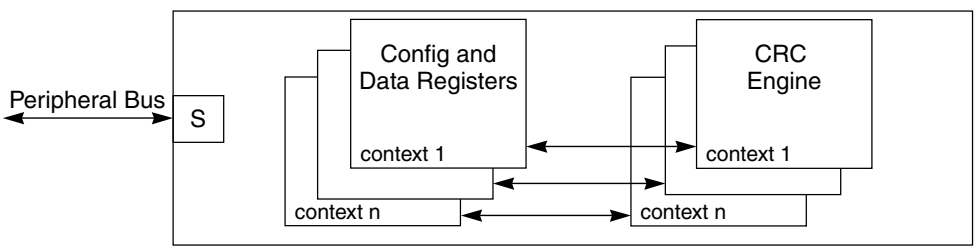


Figure 74-1. CRC top level diagram

74.4 External signal description

The CRC module does not generate any external signals.

74.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers

- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:
- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_ CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_ CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

74.5 CRC memory map and registers

CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration Register (CRC_CFG1)	32	R/W	See section	74.5.1/3648
4	Input Register (CRC_INP1)	32	R/W	0000_0000h	74.5.2/3649
8	Current Status Register (CRC_CSTAT1)	32	R/W	FFFF_FFFFh	74.5.3/3650
C	Output Register (CRC_OUTP1)	32	R	FFFF_FFFFh	74.5.4/3650
10	Configuration Register (CRC_CFG2)	32	R/W	See section	74.5.1/3648
14	Input Register (CRC_INP2)	32	R/W	0000_0000h	74.5.2/3649
18	Current Status Register (CRC_CSTAT2)	32	R/W	FFFF_FFFFh	74.5.3/3650

Table continues on the next page...

CRC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C	Output Register (CRC_OUTP2)	32	R	FFFF_FFFFh	74.5.4/3650
20	Configuration Register (CRC_CFG3)	32	R/W	See section	74.5.1/3648
24	Input Register (CRC_INP3)	32	R/W	0000_0000h	74.5.2/3649
28	Current Status Register (CRC_CSTAT3)	32	R/W	FFFF_FFFFh	74.5.3/3650
2C	Output Register (CRC_OUTP3)	32	R	FFFF_FFFFh	74.5.4/3650

74.5.1 Configuration Register (CRC_CFG*n*)

Access: User read/write.

Address: 0h base + 0h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SWAP_ BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	0	0

* Notes:

- POLYG field: Reset value is a function of n in CRC_CFGn. When n%2 = 0, reset value = 01b. When n%2 = 1,reset value = 00b. So CRC_CFG1 POLYG reset value = 00b (Since 1%2=1).

CRC_CFG*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise

Table continues on the next page...

CRC_CFG*n* field descriptions (continued)

Field	Description
	NOTE: INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8-H2F Autosar polynomial
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content.
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content

74.5.2 Input Register (CRC_INP*n*)

Access: User read/write.

NOTE

High level code must be written in a way to realize byte- , halfword- and word-wide writes on assembler code level. Any endianness problem faced can be corrected by appropriate configuration of CRC_CFG[SWAP_BYTEWISE] and CRC_CFG[SWAP_BITWISE].

Address: 0h base + 4h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_INP*n* field descriptions

Field	Description
0–31 INP	Input data for the CRC computation This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

74.5.3 Current Status Register (CRC_CSTAT*n*)

Access: User read/write.

Address: 0h base + 8h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_CSTAT*n* field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

74.5.4 Output Register (CRC_OUTP*n*)

Access: User read-only.

Address: 0h base + Ch offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_OUTP*n* field descriptions

Field	Description
0–31 OUTP	Final CRC signature This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

74.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in Equation 50 on page 3651, Equation 51 on page 3651, and Equation 52 on page 3651 for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 50. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Equation 51. CRC-CCITT (x.25 protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 52. CRC-32 (ITU-T V.42 protocol)

$$x^5 + x^3 + x^2 + x + 1$$

Equation 53. CRC-8-H2F (Autosar 4.0 protocol)

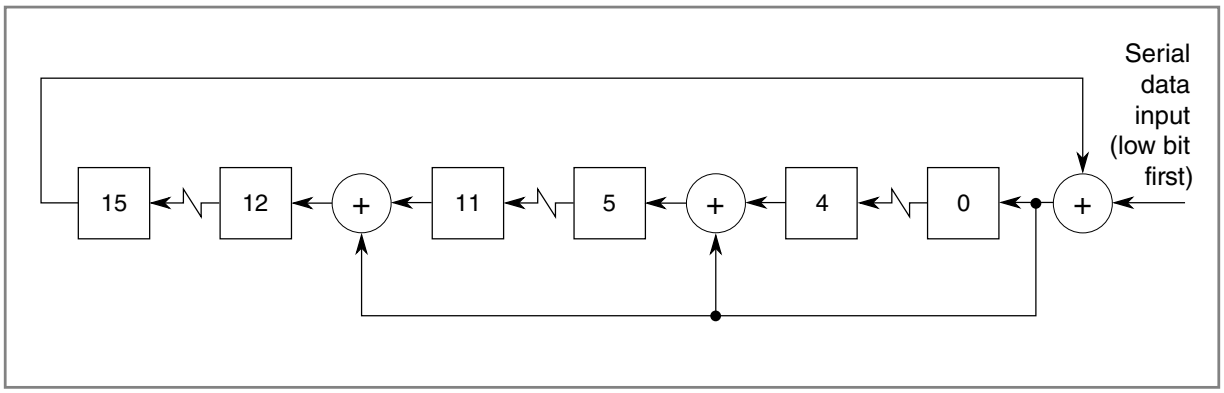


Figure 74-2. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in Figure 74-2 for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in Input Register (CRC_INP*n*), is fed first into the engine. The actual implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in Figure 74-3.

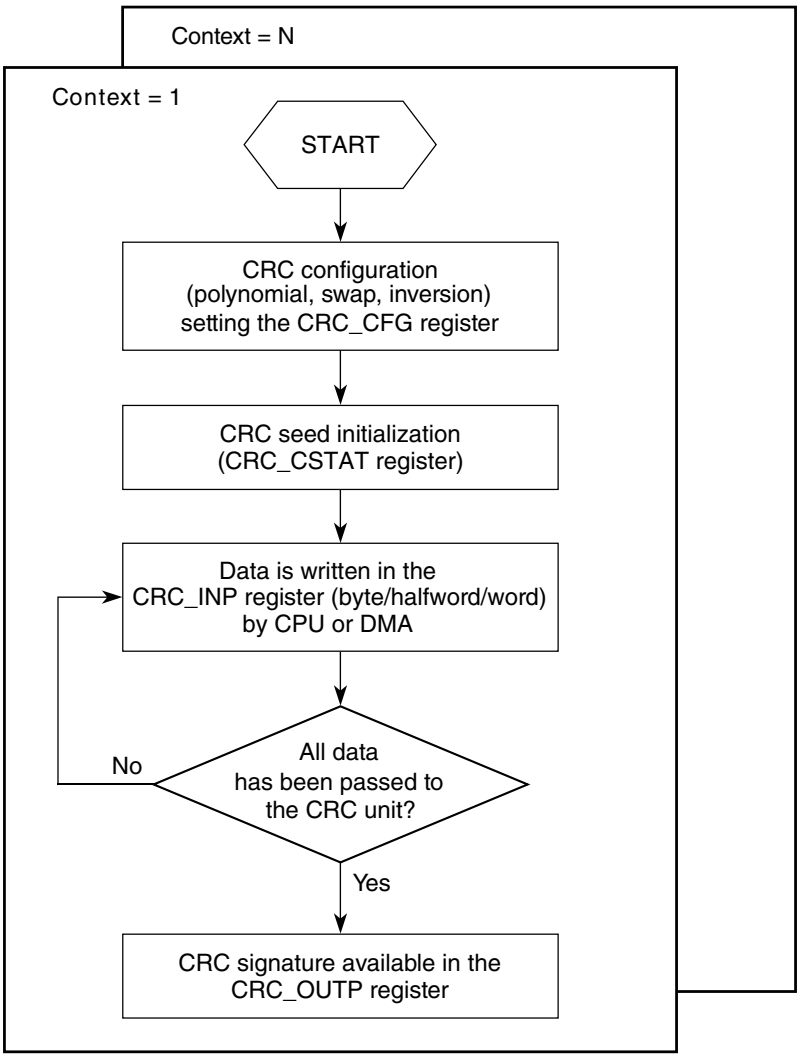


Figure 74-3. CRC computation flow

74.7 Use cases

74.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

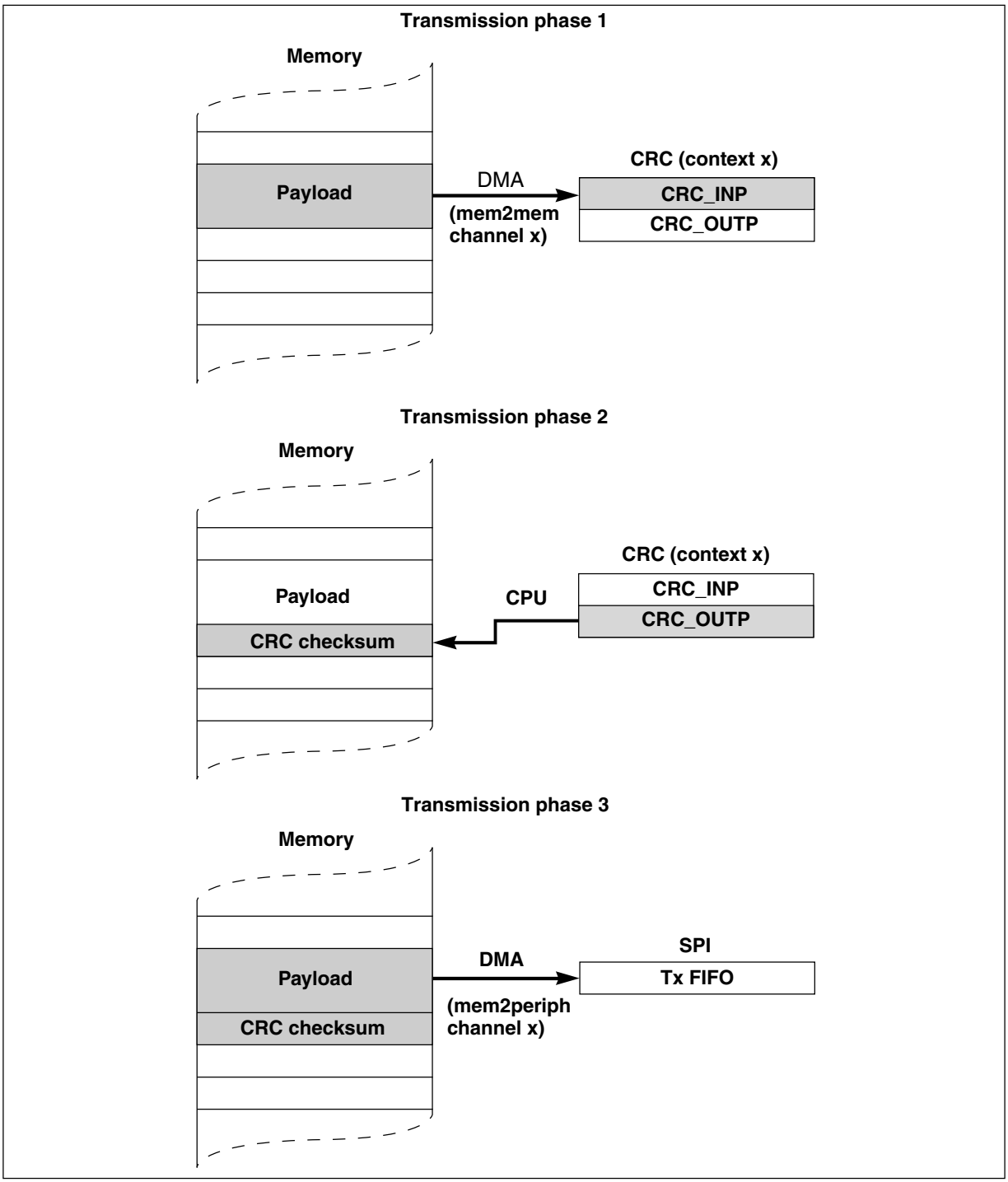


Figure 74-4. DMA-CRC transmission sequence

74.7.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the ITU-T V.42 standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 75
Memory Error Management Unit (MEMU)

75.1 Chip-specific MEMU information

75.1.1 MEMU system connections

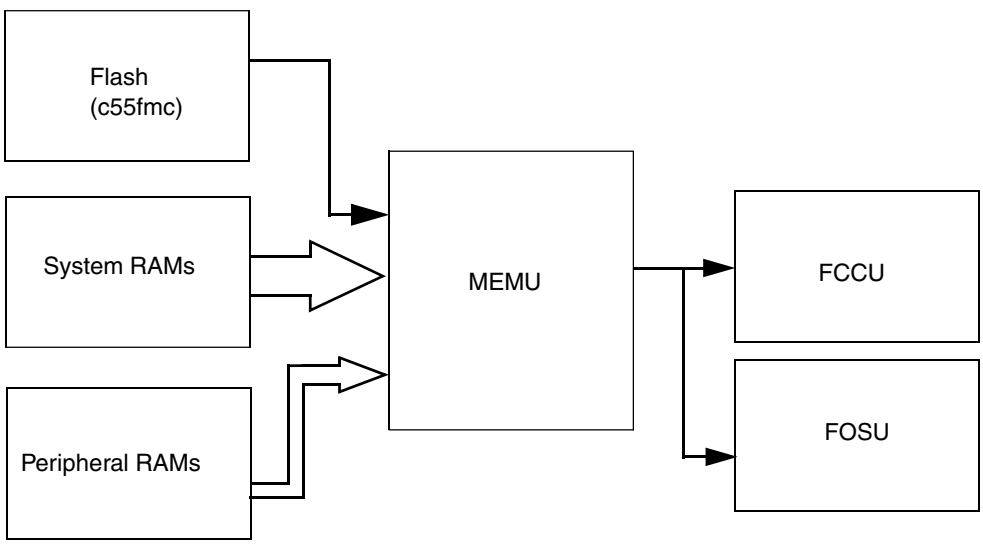


Figure 75-1. MEMU system connections

75.1.2 MEMU error sources

The following table shows the MEMU error sources for each instance configured on this chip.

Table 75-1. MEMU error sources

Instance	Value
Number of system RAM unique error sources	34

Table continues on the next page...

Table 75-1. MEMU error sources (continued)

Instance	Value
Number of peripheral RAM unique error sources	6
Number of flash memory unique error sources	3

75.1.3 Reporting tables implementation

There are 3 categories of error reporting for the MEMU:

- Flash error reporting
- System RAM error reporting—any RAM that is CPU accessible falls into this category.
- Peripheral RAM error reporting—any RAM that is not CPU accessible falls into this category.

Each category of error reporting has two tables associated with it, these are:

- Correctable Error Reporting Table
- Uncorrectable Error Reporting Table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 75-2. Number of entries in each reporting table

Error Source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash (c55fmc)	20	1
System RAM	10	1
Peripheral RAM	2	1

75.1.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from these categories:

- Flash memory
- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective field in the Concurrent Overflow registers.

Positions not explicitly listed in the tables below are reserved.

Table 75-3. Corresponding error sources for system RAM OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	z4a Data TCM (DMEM)	Asynchronous
1	z4a Instruction Cache	Asynchronous
2[1]	z4a Data Cache	Asynchronous
3	z4a Instruction AHB	Asynchronous
4	z4a Data AHB	Asynchronous
5	PRAM0 [2]	Synchronous
6	FlexRay	Synchronous
7	PBRIDGE_0	Synchronous
8	PBRIDGE_1	Synchronous
9	DMA_e2eECC	Synchronous
10	SIPI	Synchronous
11	z7a Data TCM (DMEM)	Asynchronous
12[1]	z7a Instruction Cache	Asynchronous
13[1]	z7a Data Cache	Asynchronous
14	z7a Instruction AHB	Asynchronous
15	z7a Data AHB	Asynchronous
16	z7b Data TCM (DMEM)	Asynchronous
17[1]	z7b Instruction Cache	Asynchronous
18[1]	z7b Data Cache	Asynchronous
19	z7b Instruction AHB	Asynchronous
20	z7b Data AHB	Asynchronous
21	ENET	Synchronous
22	PRAM1 [2]	Synchronous
23	PRAM2 [2]	Synchronous
24	PRAM3 [2]	Synchronous
25	PRAM4 [2]	Synchronous
26	PRAM5 [2]	Synchronous
27	PRAM6 [2]	Synchronous
28	PRAM7 [2]	Synchronous
29	SPT_e2eECC	Synchronous
30	SPT_MOD_ORRAM_Access	Asynchronous
31	SPT_PDMA_OPRAM_Access	Asynchronous

1. Cache tag errors can be single-bit or double-bit. Cache data array errors are reported as single-bit errors only, regardless of the number of errors reported, since the logic does not exist in the cache to distinguish them.

2. RAM controller does not provide a multibit ECC error on a RAM read. It will only provide the error on a read-modify-write and an error detected. If there is an error on the read, it will pass the error along to the master accessing and leave it up to that master to handle it.

Table 75-4. Corresponding error sources for system RAM OFLW1 register

Bit number	Error source	Clock relationship with respect to MEMU
0	SPT_TRAM_Access	Asynchronous
1	CSE e2e ECC gasket error	Synchronous

Table 75-5. Corresponding error sources for flash memory OFLW register

Bit number	Error source	Clock relationship with respect to MEMU
0	Flash_Core0_SPT [1]	Synchronous
1	Flash_Core1 [1]	Synchronous
2	Flash_Core2_CSE [1]	Synchronous

1. In the event of a flash ECC error, the MEMU is alerted twice - once by the flash controller and once by the master e2e_ecc gasket.

Table 75-6. Corresponding error sources for peripheral RAM OFLW register

Bit number	Error source	Clock relationship with respect to MEMU
0	FlexRay Data RAM	Asynchronous
1	FlexRay LUT RAM	Synchronous
2[1]	FlexCAN Receiver Instance 0	Synchronous
3[1]	FlexCAN Receiver Instance 1	Synchronous
4[1]	FlexCAN Receiver Instance 2	Synchronous
5	DMA_TCD_ECC	Synchronous

1. Follow below steps to decode the logical address where the ECC error occurs from the address reported by MEMU: For example, MEMU reports ECC error on address 0xFFEC0180.
    1. Extract last three nibbles from the address and shift them to left left: ((0x180<<2)= 0x600).
    2. Refer the address calculated above in the column "injection address" of the table CAN_1 Error Injection Address Register (CAN_ERRIAR) - error injection addresses and Error Injection Address Register (CAN_ERRIAR),This table is different for FD and NON-FD CAN instances.
    3. Refer memory map address corresponding to step 2 Memory map (example; RXIMR0 at 0x880).
    4. To calculate exact address, append address from Step 2 or Step 3 to first 20 bits of base address (example; 0xFFEC0 to get logical address of 0XFFEC0880)

75.1.5 FlexRay lookup table (LUT) RAM alignment

Table 75-7. Flexray Lookup Table (LUT) RAM

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register name	MEMU error reporting (PERIPH_RAM_*_ADDR)
0	FFE5_0800	FFE5_080C	FR_MBCCFR0 FR_MBFIDR0 FR_MBIDXR0 FR_MBCCFR1 FR_MBFIDR1 FR_MBIDXR1	FFE5_2100
1	FFE5_0810	FFE5_081C	FR_MBCCFR2 FR_MBFIDR2 FR_MBIDXR2 FR_MBCCFR3 FR_MBFIDR3 FR_MBIDXR3	FFE5_2101
2	FFE5_0820	FFE5_082C	FR_MBCCFR4 FR_MBFIDR4 FR_MBIDXR4 FR_MBCCFR5 FR_MBFIDR5 FR_MBIDXR5	FFE5_2102
3	FFE5_0830	FFE5_083C	FR_MBCCFR6 FR_MBFIDR6 FR_MBIDXR6 FR_MBCCFR7 FR_MBFIDR7 FR_MBIDXR7	FFE5_2103
4	FFE5_0840	FFE5_084C	FR_MBCCFR8 FR_MBFIDR8 FR_MBIDXR8 FR_MBCCFR9 FR_MBFIDR9 FR_MBIDXR9	FFE5_2104
5	FFE5_0850	FFE5_085C	FR_MBCCFR10 FR_MBFIDR10 FR_MBIDXR10 FR_MBCCFR11 FR_MBFIDR11 FR_MBIDXR11	FFE5_2105
6	FFE5_0860	FFE5_086C	FR_MBCCFR12 FR_MBFIDR12 FR_MBIDXR12 FR_MBCCFR13 FR_MBFIDR13 FR_MBIDXR13	FFE5_2106
7	FFE5_0870	FFE5_087C	FR_MBCCFR14 FR_MBFIDR14 FR_MBIDXR14 FR_MBCCFR15 FR_MBFIDR15 FR_MBIDXR15	FFE5_2107
8	FFE5_0880	FFE5_088C	FR_MBCCFR16 FR_MBFIDR16 FR_MBIDXR16 FR_MBCCFR17 FR_MBFIDR17 FR_MBIDXR17	FFE5_2108

Table continues on the next page...

Table 75-7. Flexray Lookup Table (LUT) RAM (continued)

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register name	MEMU error reporting (PERIPH_RAM_*_ADDR)
9	FFE5_0890	FFE5_089C	FR_MBCCFR18 FR_MBFIDR18 FR_MBIDXR18 FR_MBCCFR19 FR_MBFIDR19 FR_MBIDXR19	FFE5_2109
...	...	...	...	...
62	FFE5_0BE0	FFE5_0BEC	FR_MBCCFR125 FR_MBFIDR125 FR_MBIDXR125 FR_MBCCFR124 FR_MBFIDR124 FR_MBIDXR124	FFE5_213E
63	FFE5_0BF0	FFE5_0BFC	FR_MBCCFR127 FR_MBFIDR127 FR_MBIDXR127 FR_MBCCFR126 FR_MBFIDR126 FR_MBIDXR126	FFE5_213F
64	FFE5_1000	FFE5_100A	MBDOR0..5	FFE5_2140
65	FFE5_100C	FFE5_1016	MBDOR6..11	FFE5_2141
66	FFE5_1018	FFE5_1022	MBDOR12..17	FFE5_2142
67	FFE5_1024	FFE5_102E	MBDOR18..23	FFE5_2143
68	FFE5_1030	FFE5_103A	MBDOR24..29	FFE5_2144
69	FFE5_103C	FFE5_1046	MBDOR30..35	FFE5_2145
70	FFE5_1048	FFE5_1052	MBDOR36..41	FFE5_2146
71	FFE5_1054	FFE5_105E	MBDOR42..47	FFE5_2147
72	FFE5_1060	FFE5_106A	MBDOR48..53	FFE5_2148
73	FFE5_106C	FFE5_1076	MBDOR54..59	FFE5_2149
74	FFE5_1078	FFE5_1082	MBDOR60..65	FFE5_214A
75	FFE5_1084	FFE5_108E	MBDOR66..71	FFE5_214B
76	FFE5_1090	FFE5_109A	MBDOR72..77	FFE5_214C
77	FFE5_109C	FFE5_10A6	MBDOR78..83	FFE5_214D
78	FFE5_10A8	FFE5_10B2	MBDOR84..89	FFE5_214E
79	FFE5_10B4	FFE5_10BE	MBDOR90..95	FFE5_214F
80	FFE5_10C0	FFE5_10CA	MBDOR96..101	FFE5_2150
81	FFE5_10CC	FFE5_10D6	MBDOR102..107	FFE5_2151
82	FFE5_10D8	FFE5_10E2	MBDOR108..113	FFE5_2152
83	FFE5_10E4	FFE5_10EE	MBDOR114..119	FFE5_2153
84	FFE5_10F0	FFE5_10FA	MBDOR120..125	FFE5_2154
85	FFE5_10FC	FFE5_1106	MBD0R126..131	FFE5_2155
86	FFE5_1108	FFE5_1112	LEETR0..5	FFE5_2156

75.2 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- System RAM
- Peripheral RAM
- Flash memory

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single Bit error in the data part that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the chip-specific MEMU information.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from ECC logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC
- Peripheral RAM ECC
- Flash memory ECC

Each instance has two reporting tables. Correctable errors are reported in one table, and the uncorrectable errors are reported in the other.

See Design overview for the reporting block details.

75.3 Features

The MEMU has the following features:

- Supports up to 128 error sources per the following reporting blocks (the number of error sources are chip-specific; see the chip-specific MEMU information):
  - System RAM ECC
  - Peripheral RAM ECC
  - Flash memory ECC
- Support for error reporting from the following category of error sources (for ECC):
  - System RAM
  - Peripheral RAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

75.4 Block diagram

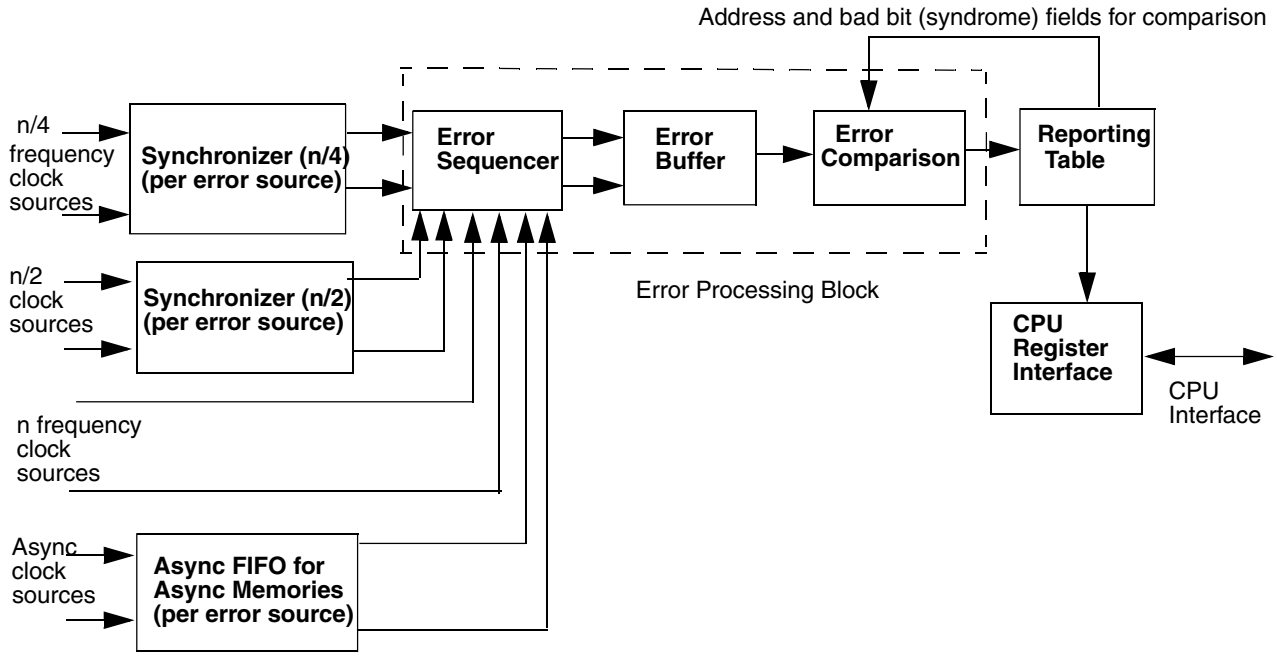


Figure 75-2. MEMU block diagram

75.5 Design overview

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See Handling overflows (Multiple error reporting), for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the

FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

This block is used to process the incoming errors coming from the device.

- Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
- All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.

  For example, the MEMU processes a 0-2-2 sequence like this:

  1. The MEMU sends the first error directly to the processing.
  2. The MEMU stores the second error in the error buffer.
  3. The MEMU performs a uniqueness check, comparing the third error with the existing entry in the error buffer for uniqueness.

  If the errors are unique, the MEMU:

  - Classifies the fourth error as an overflow
  - Sets the corresponding EBO flag in Error flag register (MEMU_ERR_FLAG)

  If the errors are not unique, the MEMU:

  - Drops the third error
  - Stores the fourth error in the error buffer
  - Does not set any EBO flag in Error flag register (MEMU_ERR_FLAG)

- While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC logic) of the incoming error with the one

stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.

- While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming error is generated from ECC logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique
- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. ..._CERR_STS or ..._UNCERR_STS). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see the chip-specific MEMU information.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

The software, in principle, is not supposed to write VLD to 1 if it is already 1.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

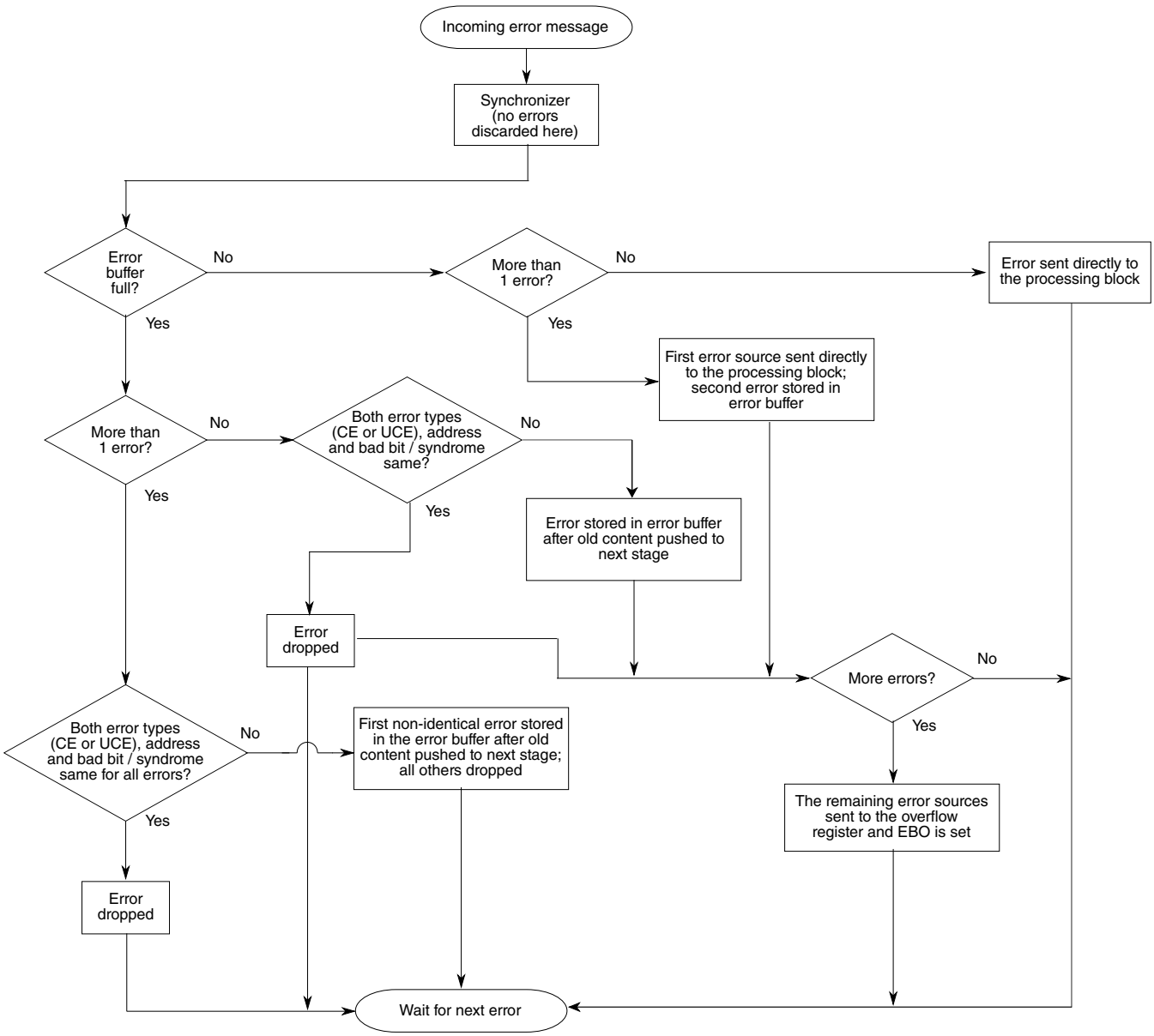


Figure 75-3. MEMU error buffer uniqueness check flow chart

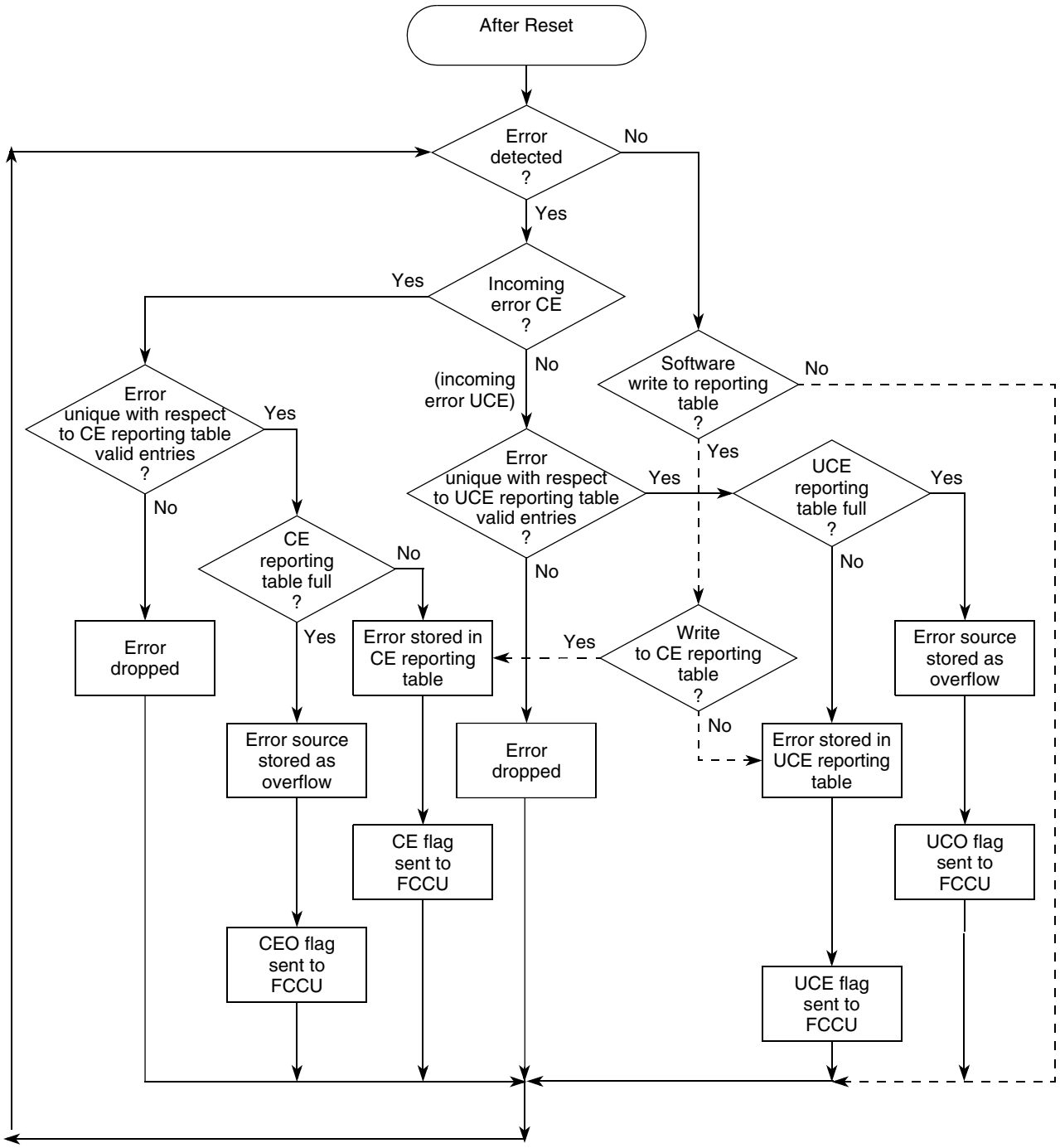


Figure 75-4. MEMU reporting table uniqueness check flow chart

75.6 External signal description

MEMU has no external signals.

75.7 Memory map and register definition

This section describes the registers on this module.

75.7.1 Overview

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

NOTE

The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

NOTE

The memory map in this section shows all the possible registers on this module and the associated offsets (which vary depending on the number of registers actually implemented). The actual availability, number and addresses of registers, and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

MEMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (MEMU_CTRL)	32	R/W	0000_0000h	75.7.2/3674
4	Error flag register (MEMU_ERR_FLAG)	32	w1c	0000_0000h	75.7.3/3674
C	Debug register (MEMU_DEBUG)	32	R/W	0000_0000h	75.7.4/3678
20	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS0)	32	R/W	0000_0000h	75.7.5/3680
24	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR0)	32	R/W	0000_0000h	75.7.6/3681
28	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS1)	32	R/W	0000_0000h	75.7.5/3680
2C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR1)	32	R/W	0000_0000h	75.7.6/3681
30	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS2)	32	R/W	0000_0000h	75.7.5/3680

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR2)	32	R/W	0000_0000h	75.7.6/3681
38	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS3)	32	R/W	0000_0000h	75.7.5/3680
3C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR3)	32	R/W	0000_0000h	75.7.6/3681
40	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS4)	32	R/W	0000_0000h	75.7.5/3680
44	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR4)	32	R/W	0000_0000h	75.7.6/3681
48	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS5)	32	R/W	0000_0000h	75.7.5/3680
4C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR5)	32	R/W	0000_0000h	75.7.6/3681
50	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS6)	32	R/W	0000_0000h	75.7.5/3680
54	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR6)	32	R/W	0000_0000h	75.7.6/3681
58	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS7)	32	R/W	0000_0000h	75.7.5/3680
5C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR7)	32	R/W	0000_0000h	75.7.6/3681
60	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS8)	32	R/W	0000_0000h	75.7.5/3680
64	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR8)	32	R/W	0000_0000h	75.7.6/3681
68	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS9)	32	R/W	0000_0000h	75.7.5/3680
6C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR9)	32	R/W	0000_0000h	75.7.6/3681
70	System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)	32	R/W	0000_0000h	75.7.7/3681
74	System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	75.7.8/3682
78	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW0)	32	w1c	0000_0000h	75.7.9/3682
7C	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW1)	32	w1c	0000_0000h	75.7.9/3682
620	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS0)	32	R/W	0000_0000h	75.7.10/ 3683
624	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR0)	32	R/W	0000_0000h	75.7.11/ 3684
628	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS1)	32	R/W	0000_0000h	75.7.10/ 3683

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
62C	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR1)	32	R/W	0000_0000h	75.7.11/ 3684
630	Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)	32	R/W	0000_0000h	75.7.12/ 3684
634	Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	75.7.13/ 3685
638	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW0)	32	w1c	0000_0000h	75.7.14/ 3685
C20	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS0)	32	R/W	0000_0000h	75.7.15/ 3686
C24	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR0)	32	R/W	0000_0000h	75.7.16/ 3687
C28	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS1)	32	R/W	0000_0000h	75.7.15/ 3686
C2C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR1)	32	R/W	0000_0000h	75.7.16/ 3687
C30	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS2)	32	R/W	0000_0000h	75.7.15/ 3686
C34	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR2)	32	R/W	0000_0000h	75.7.16/ 3687
C38	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS3)	32	R/W	0000_0000h	75.7.15/ 3686
C3C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR3)	32	R/W	0000_0000h	75.7.16/ 3687
C40	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS4)	32	R/W	0000_0000h	75.7.15/ 3686
C44	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR4)	32	R/W	0000_0000h	75.7.16/ 3687
C48	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS5)	32	R/W	0000_0000h	75.7.15/ 3686
C4C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR5)	32	R/W	0000_0000h	75.7.16/ 3687
C50	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS6)	32	R/W	0000_0000h	75.7.15/ 3686
C54	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR6)	32	R/W	0000_0000h	75.7.16/ 3687
C58	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS7)	32	R/W	0000_0000h	75.7.15/ 3686
C5C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR7)	32	R/W	0000_0000h	75.7.16/ 3687
C60	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS8)	32	R/W	0000_0000h	75.7.15/ 3686
C64	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR8)	32	R/W	0000_0000h	75.7.16/ 3687

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C68	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS9)	32	R/W	0000_0000h	75.7.15/ 3686
C6C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR9)	32	R/W	0000_0000h	75.7.16/ 3687
C70	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS10)	32	R/W	0000_0000h	75.7.15/ 3686
C74	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR10)	32	R/W	0000_0000h	75.7.16/ 3687
C78	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS11)	32	R/W	0000_0000h	75.7.15/ 3686
C7C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR11)	32	R/W	0000_0000h	75.7.16/ 3687
C80	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS12)	32	R/W	0000_0000h	75.7.15/ 3686
C84	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR12)	32	R/W	0000_0000h	75.7.16/ 3687
C88	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS13)	32	R/W	0000_0000h	75.7.15/ 3686
C8C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR13)	32	R/W	0000_0000h	75.7.16/ 3687
C90	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS14)	32	R/W	0000_0000h	75.7.15/ 3686
C94	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR14)	32	R/W	0000_0000h	75.7.16/ 3687
C98	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS15)	32	R/W	0000_0000h	75.7.15/ 3686
C9C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR15)	32	R/W	0000_0000h	75.7.16/ 3687
CA0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS16)	32	R/W	0000_0000h	75.7.15/ 3686
CA4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR16)	32	R/W	0000_0000h	75.7.16/ 3687
CA8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS17)	32	R/W	0000_0000h	75.7.15/ 3686
CAC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR17)	32	R/W	0000_0000h	75.7.16/ 3687
CB0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS18)	32	R/W	0000_0000h	75.7.15/ 3686
CB4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR18)	32	R/W	0000_0000h	75.7.16/ 3687
CB8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS19)	32	R/W	0000_0000h	75.7.15/ 3686
CBC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR19)	32	R/W	0000_0000h	75.7.16/ 3687

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CC0	Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)	32	R/W	0000_0000h	75.7.17/ 3687
CC4	Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)	32	R/W	0000_0000h	75.7.18/ 3688
CC8	Flash memory concurrent overflow register (MEMU_FLASH_OFLW0)	32	w1c	0000_0000h	75.7.19/ 3688

75.7.2 Control register (MEMU_CTRL)

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_CTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
17–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

75.7.3 Error flag register (MEMU_ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Access: User read/write

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0			SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_ERR_FLAG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: • a new Flash ECC uncorrectable error was detected. • a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SR_CE	System RAM ECC Correctable Error detect flag. Indicates that a new and unique System RAM ECCcorrectable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECC Uncorrectable Error Detect flag. Asserted when: • A new System RAM ECC uncorrectable error is detected • A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC Correctable error Overflow flag. Asserted when the System RAM ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECC Uncorrectable error Overflow flag. Asserted when the System RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC Error buffer Overflow.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Flag asserted when the internal error processing buffer of System RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

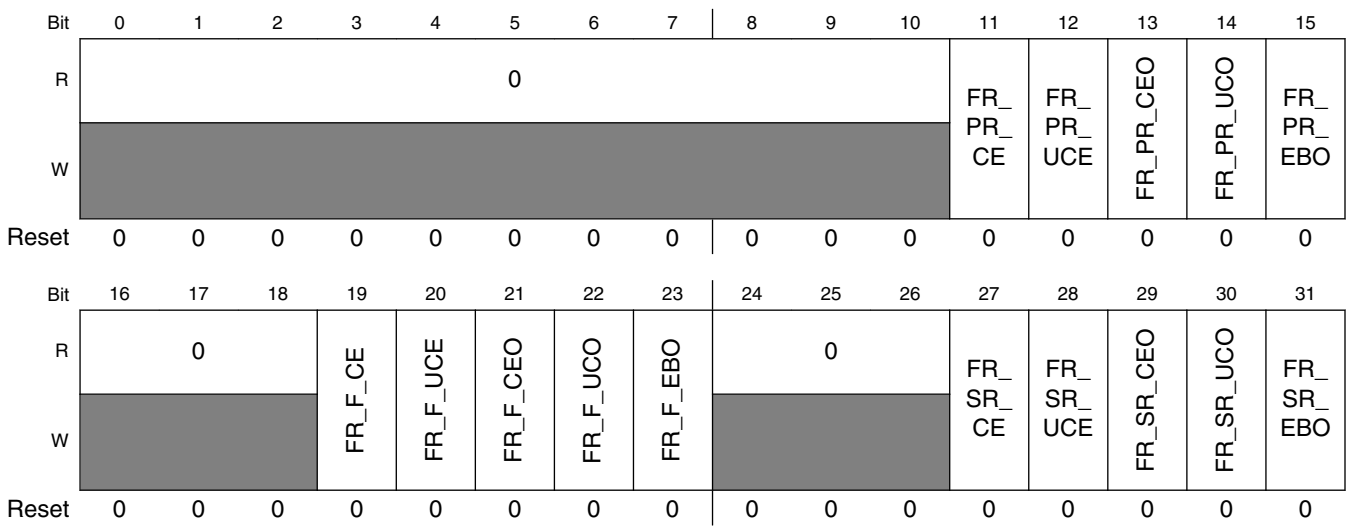
75.7.4 Debug register (MEMU_DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Access: User read/write

Address: 0h base + Ch offset = Ch



MEMU_DEBUG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

75.7.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 20h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF.

Table continues on the next page...

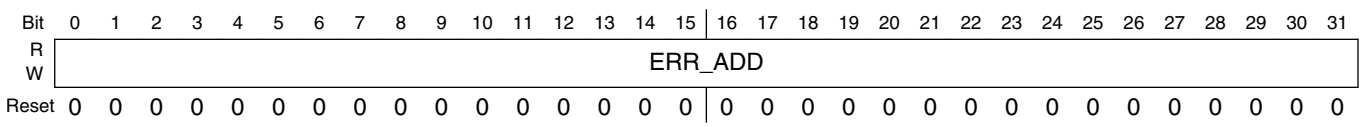
MEMU_SYS_RAM_CERR_STS*n* field descriptions (continued)

Field	Description
	Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

75.7.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 24h offset + (8d × i), where i=0d to 9d



MEMU_SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

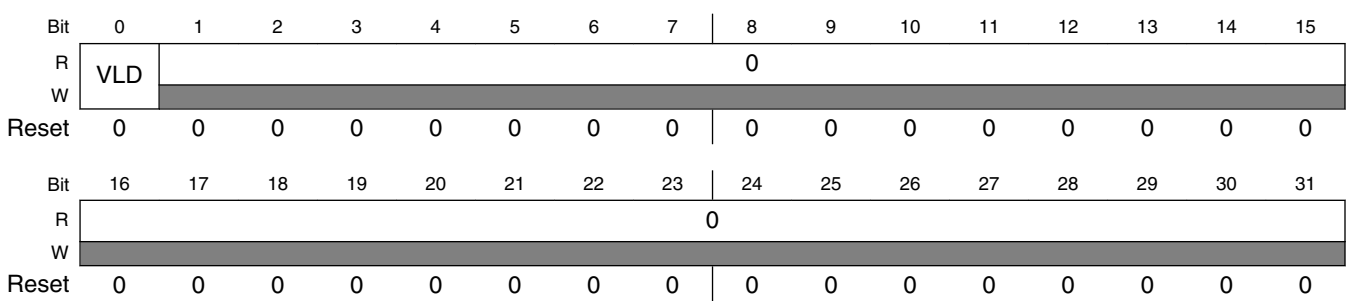
75.7.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 70h offset = 70h



MEMU_SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

75.7.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

75.7.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 78h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

75.7.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 620h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

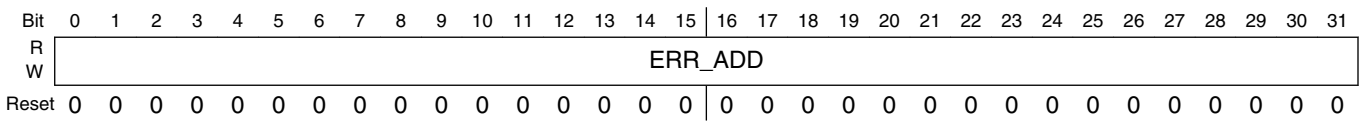
MEMU_PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

75.7.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 624h offset + (8d × i), where i=0d to 1d



MEMU_PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

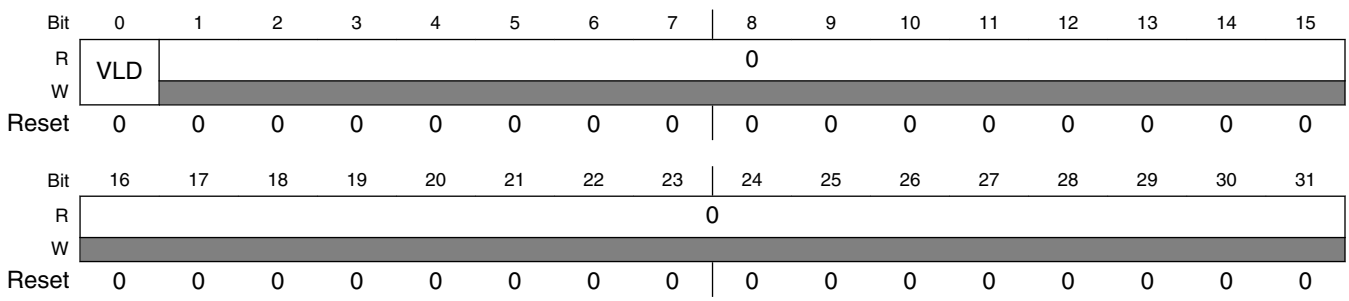
75.7.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 630h offset = 630h



MEMU_PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_PERIPH_RAM_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

75.7.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 634h offset = 634h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

75.7.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 638h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

75.7.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + C20h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

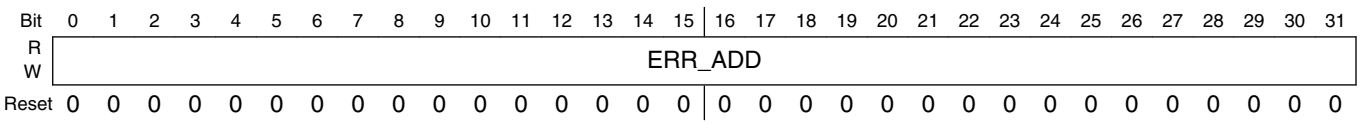
MEMU_FLASH_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid . Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in flash memory where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

75.7.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + C24h offset + (8d × i), where i=0d to 19d



MEMU_FLASH_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

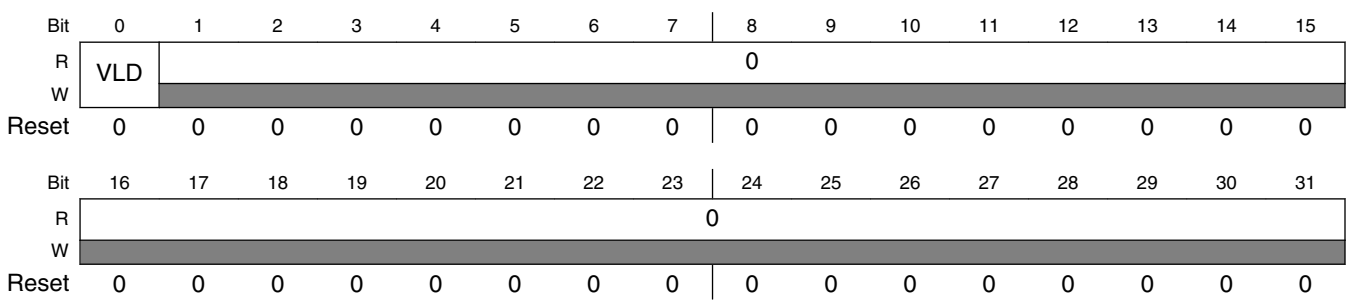
75.7.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC0h offset = CC0h



MEMU_FLASH_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_FLASH_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

75.7.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC4h offset = CC4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

75.7.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC8h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

75.8 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

75.8.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known faults which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit ='1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any fault handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes a fault with the same address that Software wants to program, both the faults will be stored in the reporting table (i.e. duplicate entries).

75.8.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

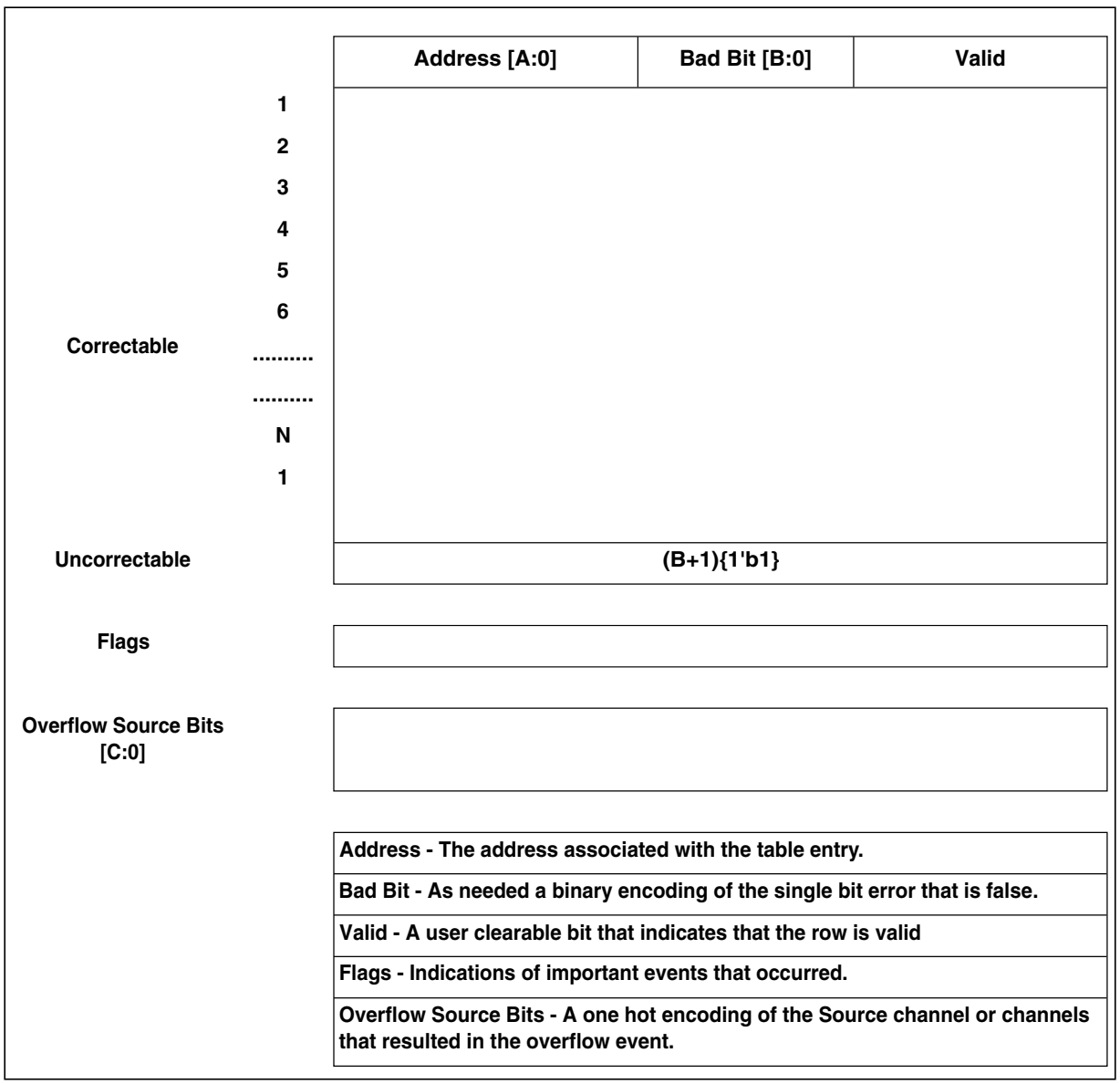


Figure 75-5. Generic representation of the reporting table of one MEMU reporting block

75.8.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Fault buffer overflow–More than two memories reporting a fault at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
    - More than 2 faults detected at same time. This would lead to overflow in the fault buffer. MEMU can process only one fault at a time while storing the other in the fault buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not "guarantee" that an ECC-supervised memory with only one fault will not be marked as overflow source. If that one fault occurs together with two (or more) additional faults at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Fault Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
3. The Overflows will be stored in the bit-wise order i.e., if the fault bit 31 reports a fault, the bit 31 of the overflow register will be set to '1'.

Chapter 76
Fault Collection and Control Unit (FCCU)

76.1 Chip-specific FCCU information

76.1.1 Fault inputs

NCF is used to identify a non-critical fault input at the FCCU. The NCF sheet attached with this document shows the source of the fault signals and the type of fault inputs to which these signals are connected at the FCCU. See the device NCF sheet (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it.

NOTE

To enable any NCF via FCCU_NCF_En, the programming sequence as followed:

- Disable all the NCF channels by writing 0000_0000h in Noncritical Fault Enable (FCCU_NCF_En)[NCFEx].
- Enable the NCF channels and its corresponding reactions.

Before exiting the CONFIG state, ensure that the NCF's which are enabled by Noncritical Fault Enable (FCCU_NCF_En) [NCFEx] have the corresponding reactions programmed.

If the above sequence is not ensured then FOSU might give destructive reset.

To recovery of enabled non-critical faults, the programming sequence for defining the preset value of the timer as followed:

- Disable all the TOEn by writing 0000_0000h in Noncritical Fault Timeout Enable(FCCU_NCF_TOEn)[NCFTOEx].
- Enable the NCFs chanel corresponding Noncritical Fault Timeout Enable(FCCU_NCF_TOEn)[NCFTOEx].and IRQ Alarm Enable (FCCU_IRQ_ALARM_ENn)[IRQENx].

NOTE

DTCM corresponds to DMEM. For details, see Core chapters in this document.

76.1.1.1 Core lockstep and RCCU disablement in debug mode

In debug mode, the RCCUs are disabled, and the cores do not operate in Lockstep mode.

The actual enabled/disabled state of the RCCUs (including any automatic disablement by debug) can be read by software in the MCB_MISC2[RCCU_Z4_STATUS].

Table 76-1. MCB_MISC2 register details

Address	Register name	Width (in bits)	Access	Reset value
FFC0_C014h	MCB_MISC2	32	RW	0000_0000h

NOTE

When the Z4 Core enters into Debug Mode, DMA RCCU is also disabled along with the Z4 RCCU. The status of DMA RCCU can be read from MCB_MISC2[RCCU_DMA_STATUS].

76.1.2 Available FCCU output signals

The following table shows the available FCCU output signals implemented on this chip.

Table 76-2. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0][1]	FCCU_F[0]
EOUT[1]	FCCU_F[1]

1. During first configuration phase EOUT pads will be in high-Z state.

76.1.3 FCCU chip-specific register reset values

This table lists in alphabetical order by mnemonic the FCCU registers whose reset values are specific to this chip. (The reset values for all other FCCU registers are specified in Register descriptions.)

Register	Reset value
FCCU_CFG	See FCCU_CFG register bit value sources (N and C) by event.
FCCU_CFG_TO	0000_0006h
FCCU_NCF_CFG0	FFFF_FFFFh
FCCU_NCF_CFG1	FFFD_FFFFh
FCCU_NCF_CFG2	FFEF_FCCFh
FCCU_NCF_CFG3	FFFF_FFFFh
FCCU_NCF_E0	FFFF_FFFFh
FCCU_NCF_E1	0000_3FFFh
FCCU_NCF_E2	FF80_00C0h
FCCU_NCF_E3	FFFF_FFFFh
FCCU_NCF_TO	0000_1F00h
FCCU_NCF_TOE0	FFFF_FFFFh
FCCU_NCF_TOE1	FFFD_FFFFh
FCCU_NCF_TOE2	FFEF_FCCFh
FCCU_NCF_TOE3	FFFF_FFFFh
FCCU_NCFS_CFG0	0000_0000h
FCCU_NCFS_CFG1	0000_0000h
FCCU_NCFS_CFG2	0000_0000h
FCCU_NCFS_CFG3	0000_0000h
FCCU_NCFS_CFG4	0000_0000h
FCCU_NCFS_CFG5	0000_0000h
FCCU_NCFS_CFG6	0000_0000h
FCCU_NCFS_CFG7	0000_0000h
FCCU_MCS	0000_8083h (after reset the chip enters to the DRUN mode).

76.1.4 FCCU_CFG register event bit values by source (N and C)

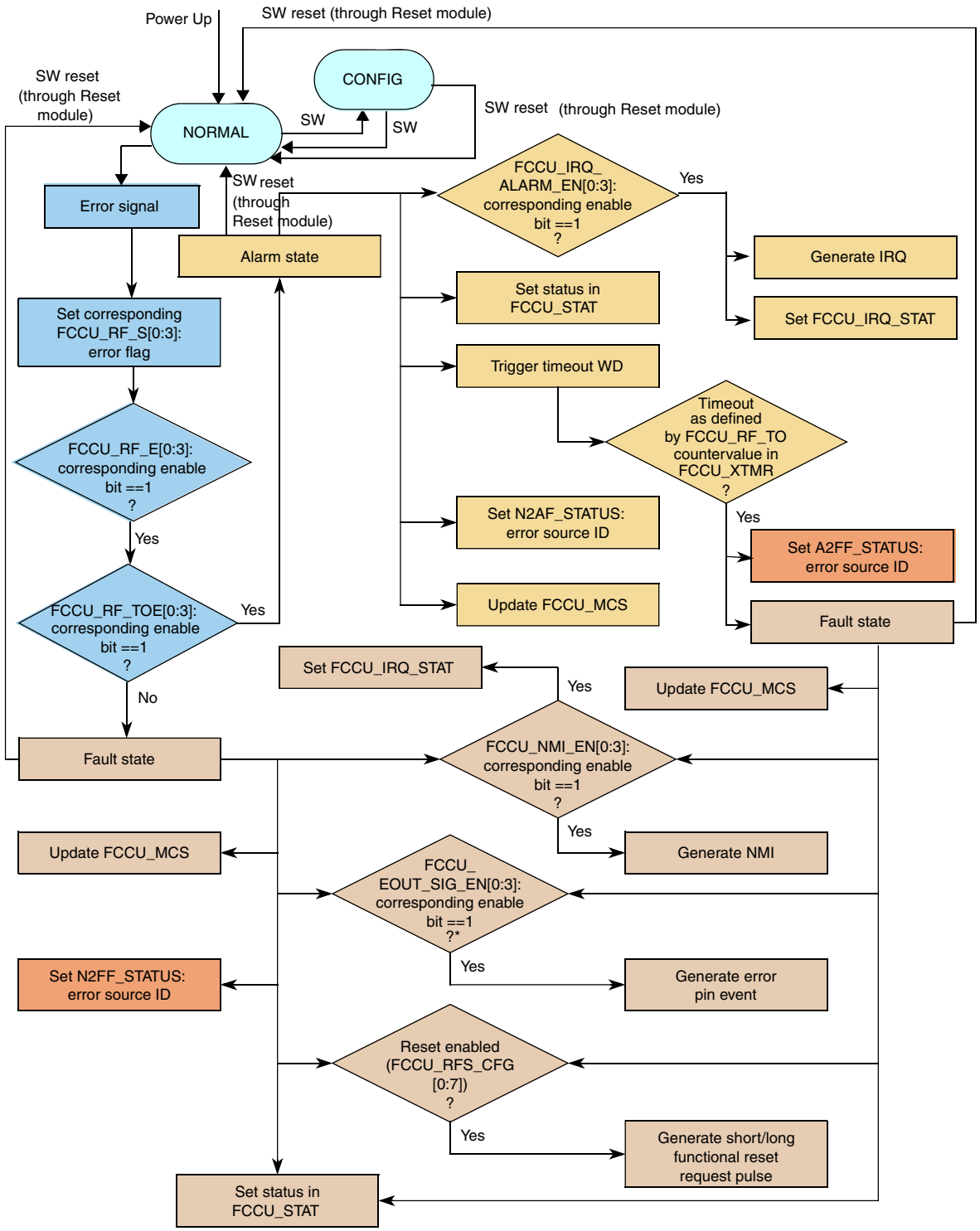
For each bit in the Configuration (FCCU_CFG) register, this table specifies the values of *N* and *C* for this chip. For information on how to interpret and use these values, see FCCU_CFG register bit value sources (N and C) by event.

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
0	0	0
1	0	0
2	0	0

Table continues on the next page...

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
3	0	0
4	0	0
5	0	0
6	0	0
7	—	0
8	—	0
9	—	0
10	—	0
11	—	0
12	—	0
13	—	0
14	—	0
15	—	0
16	0	0
17	0	0
18	0	0
19	—	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	1	0
27	1	0
28	1	1
29	1	0
30	1	0
31	1	0

76.1.5 Fault signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 76-1. Fault signal flow

76.1.6 FCCU FOSU Count Value

The fixed preprogrammed value for FOSU_Count is 0x1F40.

76.1.7 FCCU nvm_cfg-interface signal functions

Please refer to dcl_fccu in the attached DCF sheet for the description of the various the functions of the nvm_cfg signal interface of FCCU.

76.1.8 Clock sources

This table specifies the internal chip clock that is connected to each FCCU clock input.

Table 76-3. FCCU clock sources

FCCU clock input	Internal chip clock
CLKSYS	PBRIDGE_1_CLK
CLKSAFE0	IRC_CLK
CLKSAFE1	IRC_CLK

76.2 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of hardware checker (for example, RCCU) results
- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
  - No reaction

    - IRQ
    - Short functional reset
    - Long functional reset
    - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- Lockable configuration
- Nonvolatile memory interface (NVMCFG signals) for configuration loading
- One of the fault-output (EOUT) signals is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power on, the EOUT signals have high impedance.[1] They show operational state only after configuration by software.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles**.**

The FCCU circuitry is checked at the start-up by the self-checking procedure. The FCCU is operative with the default configuration immediately after the completion of the self-checking procedure. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

76.2.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out

Table continues on the next page...

1. Actual value depends on device specific setting at pad level.

Term	Description
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC_ME	Mode entry module
NCF	Noncritical fault
NMI	Nonmaskable interrupt
NVM	Nonvolatile memory
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

76.3 Main features

- Management of noncritical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
  - long functional reset request pulse
  - short functional reset request pulse
  - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock
- Nonvolatile memory interface (NVMCFG signals) for configuration loading

76.4 Block diagram

The following figure is a top-level diagram of the FCCU module.

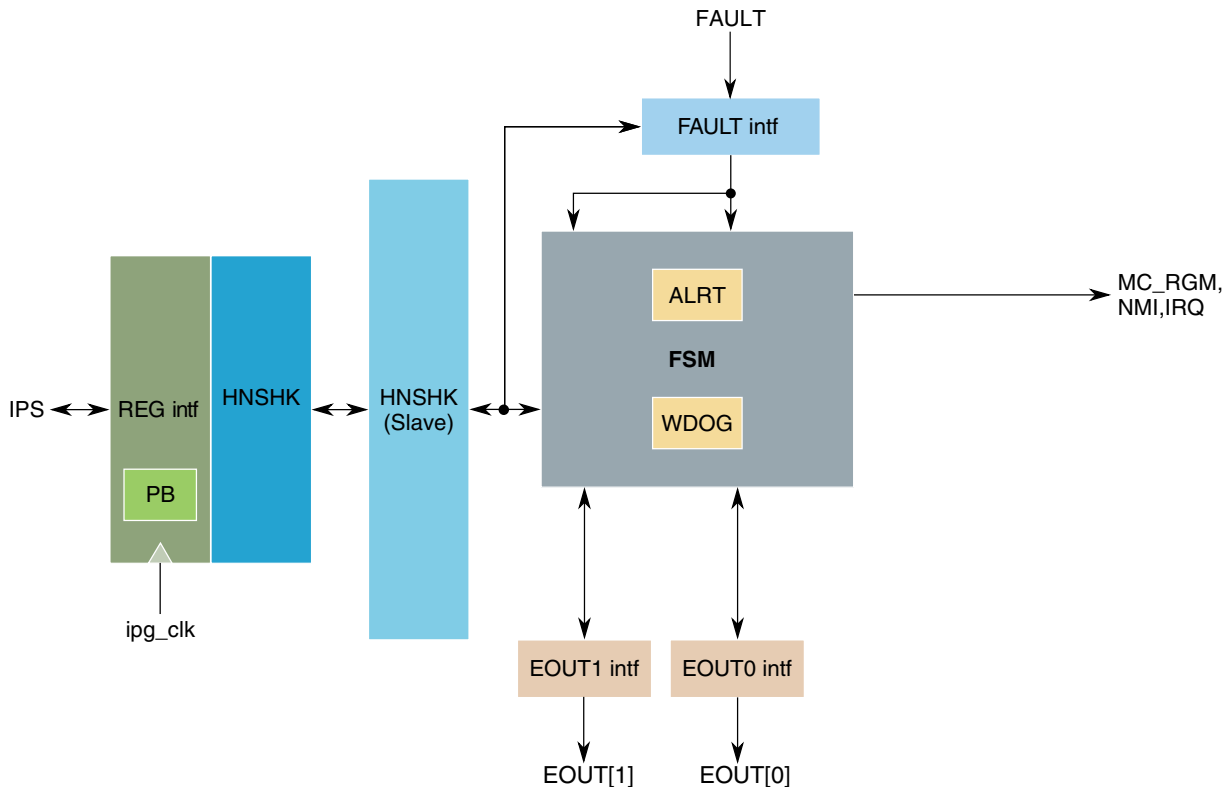


Figure 76-2. FCCU block diagram

This table describes the FCCU submodules.

Table 76-4. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

76.5 Signal description

76.5.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EIN[1:0]	Input	Error Input—Provide the mechanism for FCCU to capture the states that off-chip logic drives on the associated EOUT[1:0] signals. (On the chip, the EOUT signals are connected to their corresponding EIN signals.) FCCU captures the states in the EIN1 and EIN0 fields.	—	Asynchronous
EOUT[1:0]	Output	Error Output—Indicate FCCU's condition (Faulty, Nonfaulty, or Configuration) to off-chip logic.	Programmable	CLKSAFE0, CLKSAFE1
FOSU interface				
FIF	Output	FCCU In Fault—Indicates to the FOSU module, if present, that FCCU is recognizing a fault. FCCU asserts and then deasserts this signal in two cases: *Case 1* When the EOUT signals are programmed to be always low (FCCU_SET_CLEAR), FCCU asserts this signal until the EOUT signals are programmed not to be always low. *Case 2*	High	CLKSAFE0, CLKSAFE1

Table continues on the next page...

Signal	Direction	Function	Active level	Clock
		When the EOUT signals are programmed not to be always low (FCCU_SET_CLEAR), FCCU asserts this signal when all of the following are true: • FCCU enters Fault state as the result of a fault. • If the fault is a noncritical fault, and the EOUT signals are in a fault-output mode (FOM) other than Bistable, the associated EOUTENx field is set to enabled. FCCU deasserts this signal when FCCU leaves Fault state.		
WKPU interface				
NMIWKPU	Input	WKPU Nonmaskable Interrupt Request—Receives nonmaskable interrupt requests from the WKPU module.	Low	CLKSYS
WAKEWKPU	Output	Wake WKPU—Wakes the WKPU module to bring the chip out of dormant state when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS
NVM interface				
NVMCFG[21:0]	Input	Nonvolatile Memory Configuration—Provide the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.	Static value	—
SSCMDONE	Input	SSCM Done—After an FCCU reset, indicates that the SSCM module is now driving any NVMCFG signals	High	CLKSYS

Signal	Direction	Function	Active level	Clock
		that are associated with a data record stored in nonvolatile (flash) memory.		

76.6 Put FCCU in Configuration or Normal state

76.6.1 Introduction

You put FCCU in Configuration state to change its configuration. You put FCCU in Normal state to save changes to the configuration.

After you configure FCCU, you can lock its configuration to prevent accidental changes.

76.6.2 About changing states

When changing states, keep the following in mind:

- To put FCCU in Configuration state, FCCU must be in Normal state.
- After FCCU is reset, the configuration is temporarily locked. You must temporarily unlock the configuration before you can put FCCU in Configuration state.
- When FCCU is in Configuration state, FCCU doesn't actually save the changes you make to the configuration. To save changes to the configuration, you must manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers.

76.6.3 About locking the configuration

When locking the configuration, keep the following in mind:

- If you attempt to lock the configuration while FCCU is in Configuration state, FCCU doesn't actually lock the configuration until FCCU leaves Configuration state—that is, either you put FCCU in Normal state, or FCCU puts FCCU in Normal state because the configuration-timer interval (TO) expires.
- After you permanently lock the configuration, you must reset FCCU before you can put FCCU in Configuration state.

76.6.4 Put FCCU in Configuration state

1. From Supervisor mode, temporarily unlock the configuration (TRANSKEY).
2. To determine FCCU's state, run the OP3 operation (see Run an operation).
3. Check the operation status (OPS).
   - If the operation status is Successful, go to step 4.
   - If the operation status is Aborted, go to step 2.
4. Check the FCCU status (STATUS).
   - If the FCCU status is Configuration, FCCU is in Configuration state. Stop.
   - If the FCCU status is Normal, go to step 5.
   - If the FCCU status is Alarm or Fault, go to step 7.
5. To put FCCU in Configuration state, run the OP1 operation (see Run an operation).
6. Check the operation status (OPS).
   - If the operation status is Successful, FCCU is in Configuration state. Stop.
   - If the operation status is Aborted, the configuration is probably permanently locked. Go to step 9.
7. Recover all faults.
8. Go to step 2.
9. Reset FCCU.

76.6.5 Put FCCU in Normal state

1. To determine FCCU's state, run the OP3 operation (see Run an operation).
2. Check the operation status (OPS).
   - If the operation status is Successful, go to step 3.
   - If the operation status is Aborted, go to step 1.
3. Check the FCCU status (STATUS).
   - If the FCCU status is Normal, FCCU is in Normal state. Go to step 8.
   - If the FCCU status is Configuration, go to step 4.
   - If the FCCU status is Alarm or Fault, go to step 6.
4. To put FCCU in Normal state, run the OP2 operation (see Run an operation).
5. Check the operation status (OPS).

- If the operation status is Successful, FCCU is in Normal state. Go to step 8.
- If the operation status is Aborted, go to step 4.

6. Recover all faults.
7. Go to step 1.
8. If you want to prevent changes to the configuration, lock it:
   - To require a key to unlock the configuration, from Supervisor mode, temporarily lock the configuration (TRANSKEY).
   - To require a reset of FCCU to unlock the configuration, from Supervisor mode, permanently lock the configuration (PERMNTKEY).

     The configuration is permanently locked until FCCU is reset.

76.7 Run operations

76.7.1 Introduction

You run operations to perform actions such as putting FCCU in Configuration state or reading a timer. For a complete list of operations you can run, see OPR.

76.7.2 About running operations

When running operations, keep the following in mind:

- FCCU ignores any operations initiated while the operation status is In Progress.
- Certain operations must be unlocked before you can initiate them. After you initiate them, they are again locked.

76.7.3 Run an operation

1. Check the operation status (OPS).
2. If the operation status is In Progress, go to step 1.
3. If the operation must be unlocked before you can initiate it, unlock the operation (CTRLK) using a 32-bit write.

NOTE

The Control Key (FCCU_CTRLK) register used in this step and the Control (FCCU_CTRL) register used in the next

step must be written with consecutive instructions. Do not use read-modify-write instructions, such as bit-field instructions, to modify these registers.

4. Initiate the operation (OPR) using a 32-bit write.
5. Check the operation status (OPS).
6. If the operation status is In Progress, go to step 5.

The operation status is now Idle (OP15 only), Aborted, or Successful.

76.8 Functional description

76.8.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

These type of chip resets reset the FCCU:
- Power-on
- Destructive
- Long functional, when initiated by the assertion of the chip reset pin (RESET_B)

These types of chip resets don’t reset the FCCU:
- Long functional, when initiated by anything other than the assertion of the chip reset pin (RESET_B)
- Short functional reset

For more information on each type of reset, see the MC_RGM and reset chapters.

76.8.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- Configuration: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

  Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU., The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

  After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

  The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

  The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

  The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

  All pending faults that occur during Configuration state result in both of the following:

  - Highest-priority state transition
  - Interrupt generation (NMI or alarm IRQ)

If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:
  - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
  - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
  - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.
- Fault: FCCU moves into Fault state when one of the following condition occurs:
  - timeout related to a noncritical fault when FCCU is in Alarm state
  - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed to all cores.

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

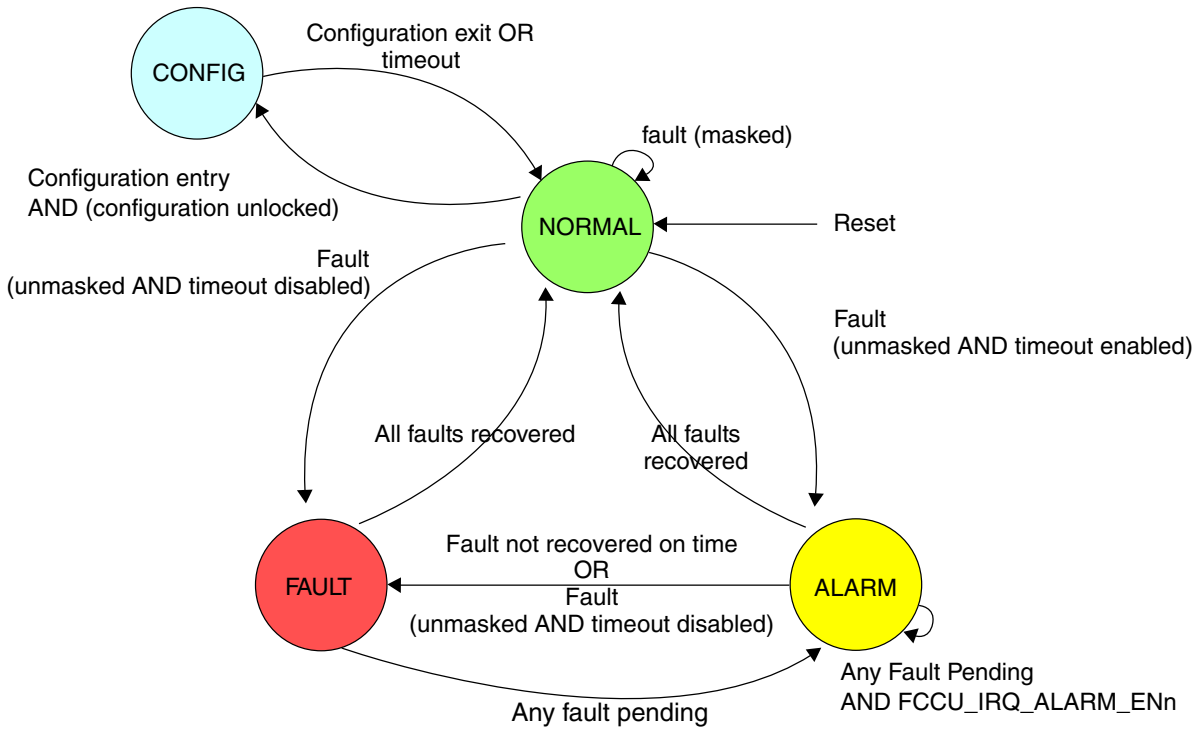


Figure 76-3. FCCU state diagram

76.8.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (noncritical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a noncritical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any noncritical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the noncritical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

76.8.4 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a noncritical FAULT management (ALARM state), see Figure 76-4 and Figure 76-5, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)
  - FCCU state transition ALARM → NORMAL

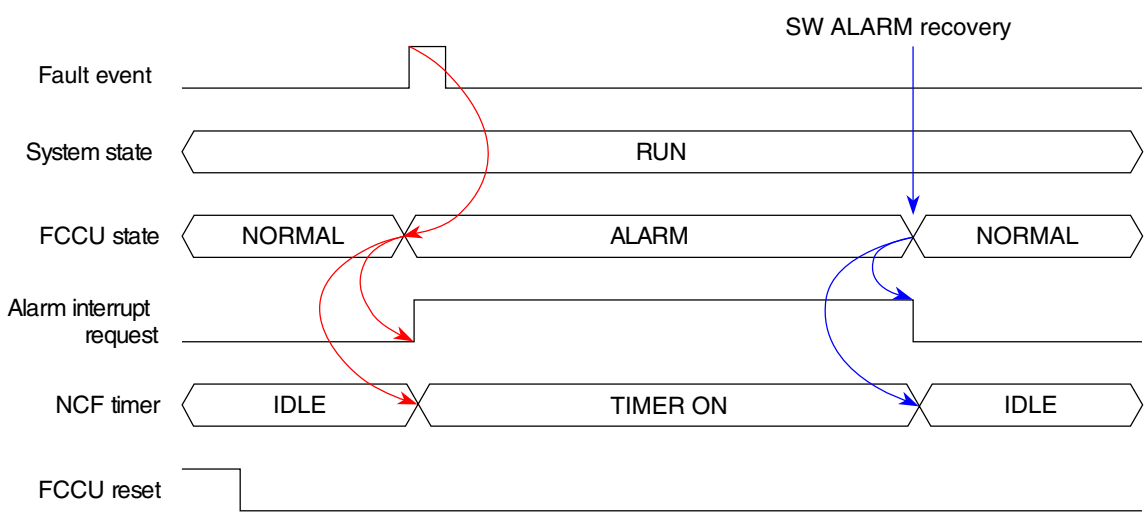


Figure 76-4. Noncritical FAULT (ALARM state) recovery (a)

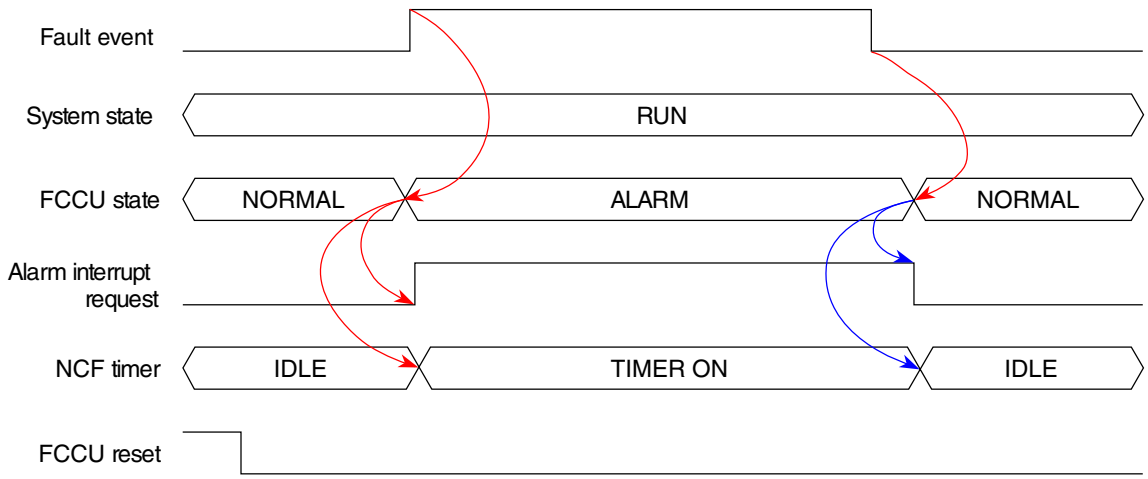


Figure 76-5. Noncritical FAULT (ALARM state) recovery (b)

A typical sequence related to a noncritical FAULT management (ALARM → FAULT state), see Figure 76-6, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
  - NMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
  - system state transition: SAFE → RUN

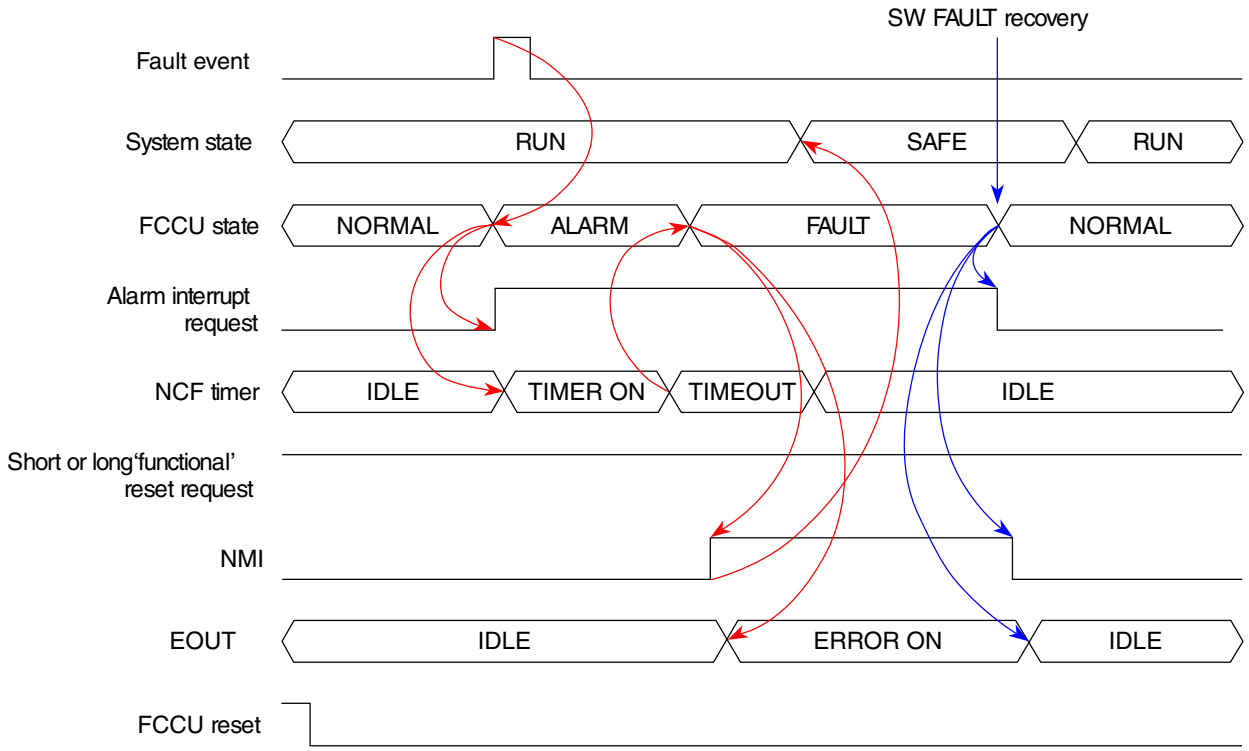


Figure 76-6. FAULT (ALARM → FAULT state) recovery

76.8.5 NMI/WKPU interface

NMIOUT is connected to all processor cores on the chip but is masked when any one of the following is true:

- An FCCU asynchronous reset is in progress.
- The chip is in RESET mode.
- The chip has automatically entered DRUN mode from RESET mode and software has not yet requested another mode change.

This figure shows the logical scheme.

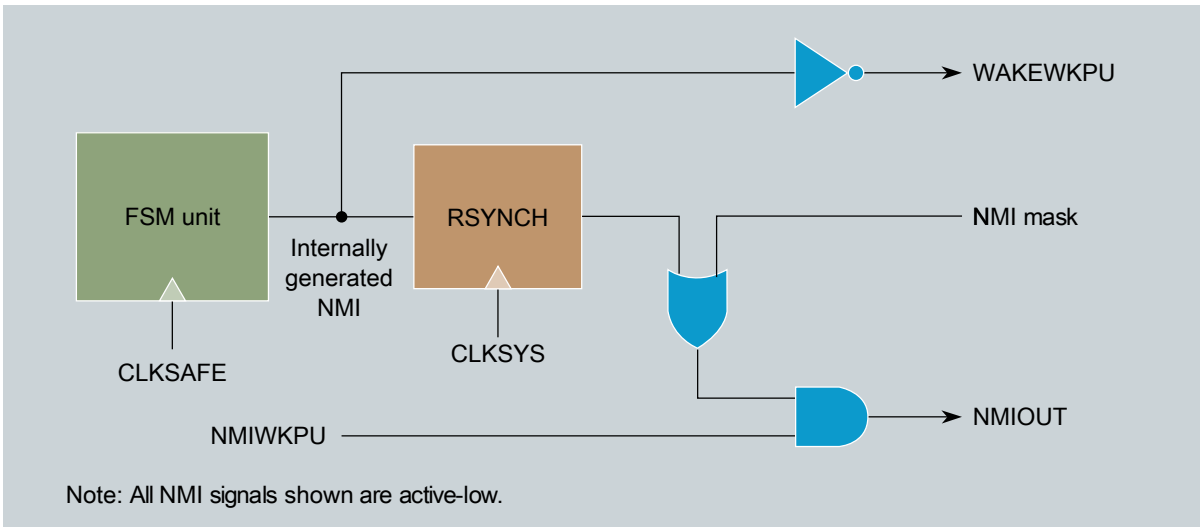


Figure 76-7. NMI/WKPU scheme

76.8.6 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable using MC_RGM.

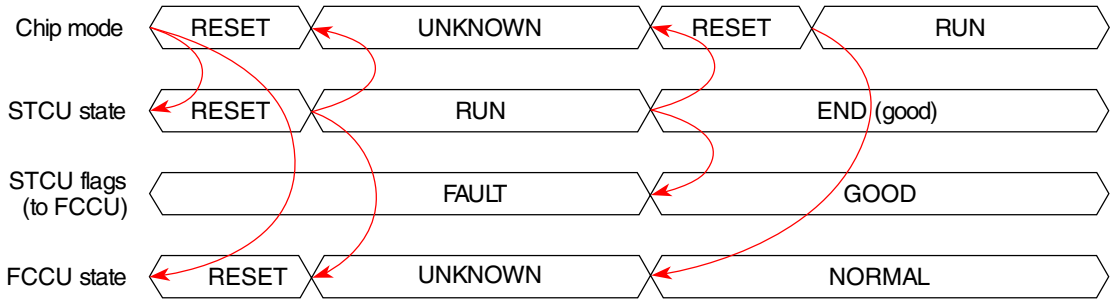


Figure 76-8. STCU-FCCU (case 1)

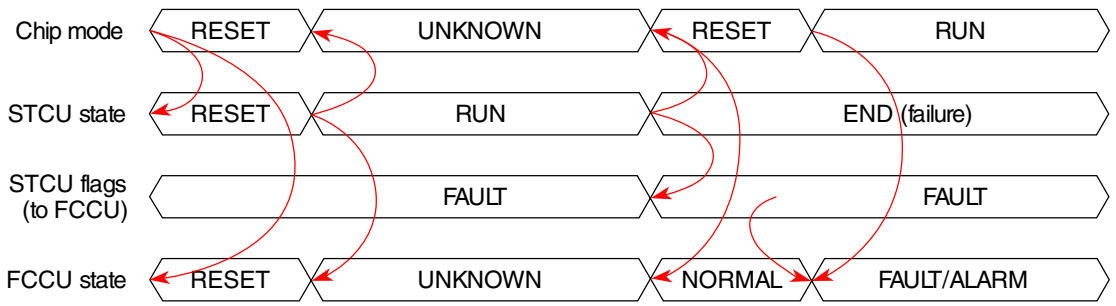


Figure 76-9. STCU-FCCU (case 2)

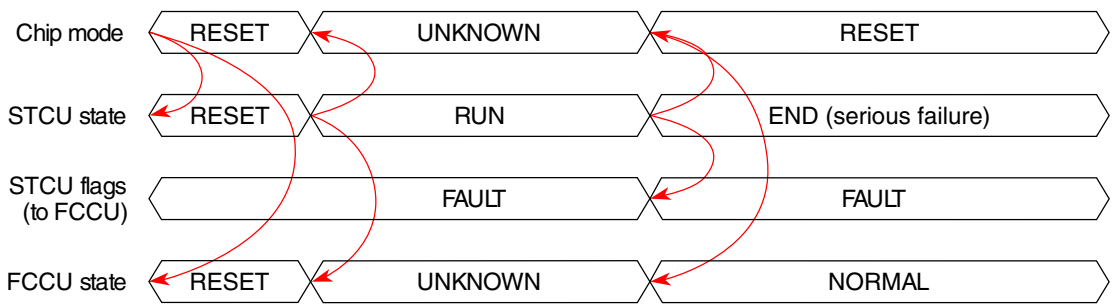


Figure 76-10. STCU-FCCU (case 3)

76.8.7 Nonvolatile memory interface

FCCU includes a nonvolatile memory interface (NVMCFG signals) that provides the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Whether any of these signals are actually connected to nonvolatile (flash) memory is chip-specific. Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.

The timing diagram related to the NVM interface is given in Figure 76-11.

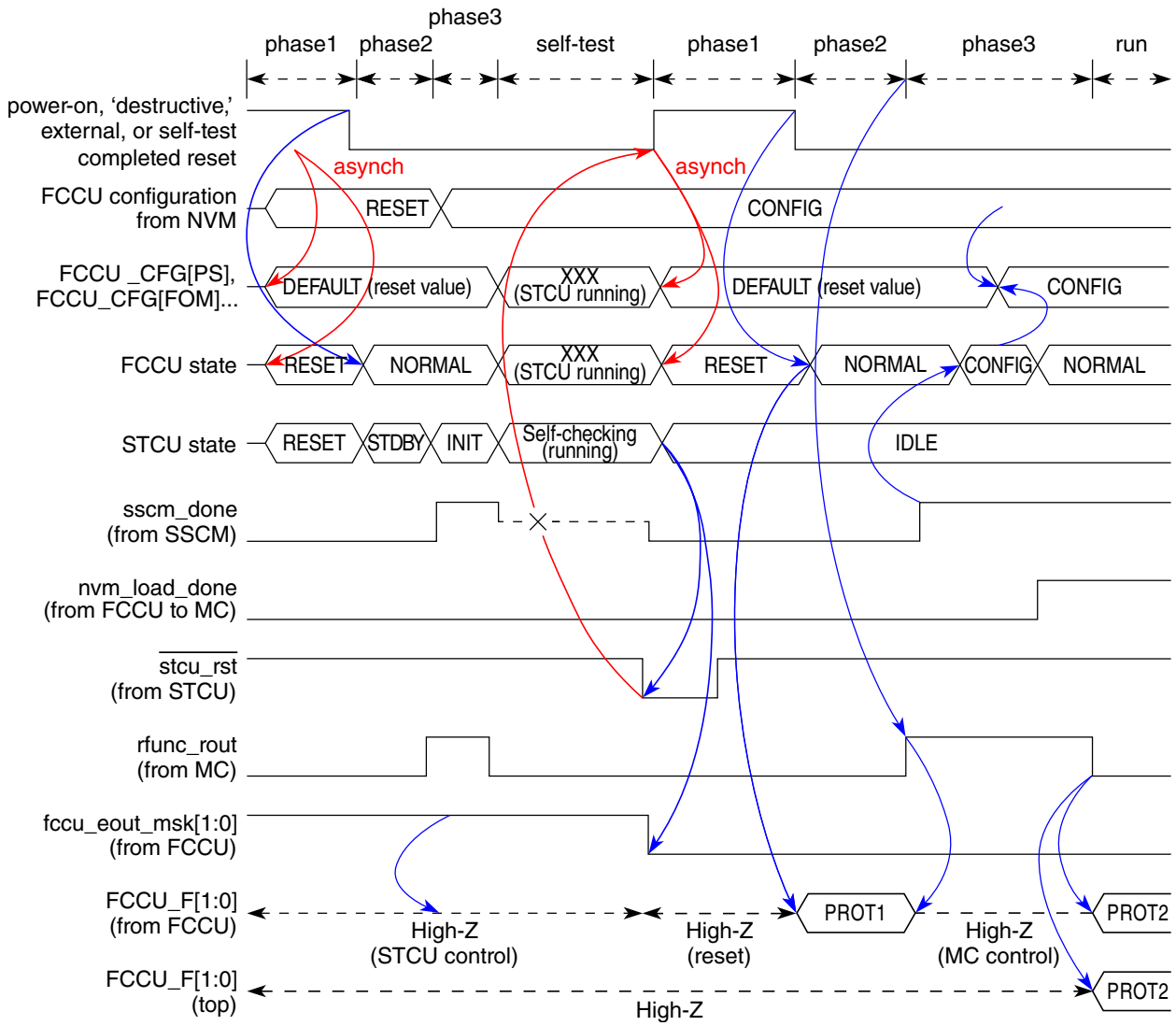


Figure 76-11. NVM interface

76.8.8 EOUT interface

76.8.8.1 Introduction

You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

76.8.8.2 The FCCU conditions

There are three FCCU conditions:

Condition	Description
Faulty	The fault-output (EOUT) timer is running (see How the fault-output (EOUT) timer works).
Nonfaulty	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Alarm or Normal state.
Configuration	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Configuration state.

76.8.8.3 How the fault-output (EOUT) timer works

FCCU starts the fault-output (EOUT) timer when all of the following are true:

- The fault-output (EOUT) signals are not programmed to be always low (FCCU_SET_CLEAR).
- The fault-output (EOUT) timer isn't already running.
- FCCU enters Fault state as the result of a fault.

When the fault-output (EOUT) timer is already running and a new fault occurs:

- When FCCU is in Configuration state: FCCU doesn't restart the fault-output (EOUT) timer.
- When FCCU is in Normal state or Alarm state:
  - And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the fault-output (EOUT) timer.
  - And Alarm state is disabled for the fault (noncritical): FCCU enters Fault state and restarts the fault-output (EOUT) timer.
- When FCCU is in Fault state: FCCU restarts the fault-output (EOUT) timer.

FCCU stops and reinitializes the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bistable fault-output mode (FOM), T_min has expired.
- All faults that caused FCCU to enter or remain in Fault state since FCCU started the fault-output (EOUT) timer have been cleared, causing FCCU to return to Normal state.

76.8.8.4 Prepare the EOUT signals to indicate FCCU's condition

Make sure all of the following are true:

- The EOUT signals are active (FCCU_SET_AFTER_RESET).
- The EOUT signals are controlled by FCCU's finite state machine (FCCU_SET_CLEAR).

76.8.8.5 More about the EOUT interface

Because SIUL2 does not control the pad-buffer enable for an Alternate function, FCCU provides a mechanism to drive the pads for EOUT[1:0] in Open-Drain (OD) mode (FCCU_CFG[OD]) so that a pad is in a high-impedance state when FCCU drives a logic 1 and is low when FCCU drives a logic 0.

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching
- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.
Derived prescaler	FCCU_CFG[FOPE], FCCU_CFG[FOP]	0

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_min, even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK,

such as an incoming fault is not masked when software has set the error signal to high. During the T_min by a non-software fault, the FCCU FSM moves independently of this signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, *NA* (not applicable) appears in the following table.

NOTE

The following table is valid only when the EOUT signals are active (FCCU_SET_AFTER_RESET).

Table 76-5. FCCU error pin behavior on state transitions

Fault signaling state	Transition from Normal to Fault state			Normal+Configuration states			Transition from Normal to Alarm state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

76.8.8.6 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 76-6 and the related timing diagram is given in Figure 76-12 and Figure 76-13.

Table 76-6. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

In the RESET phase the output pins are set as "high impedance".

Note

Figure 76-12 and Figure 76-13 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

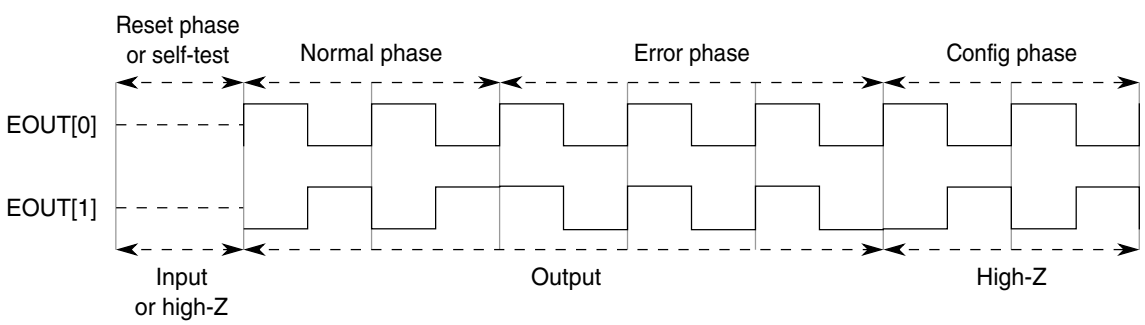


Figure 76-12. Dual-rail protocol (slow switching mode)

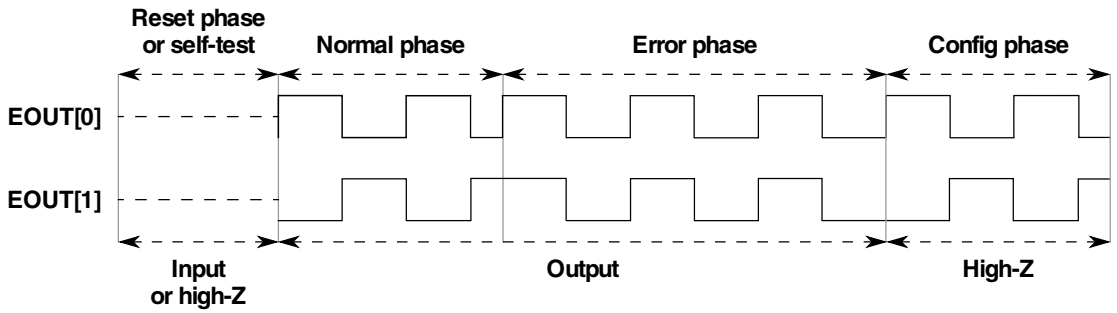


Figure 76-13. Dual-rail protocol (fast switching mode)

76.8.8.7 Time-switching protocol

The encoding scheme is given in Table 76-7 and the related timing diagram is given in Figure 76-14.

Table 76-7. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency specified by the CFG[FOP].

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 76-14 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

2. Actual value depends on device specific settings at pad level.

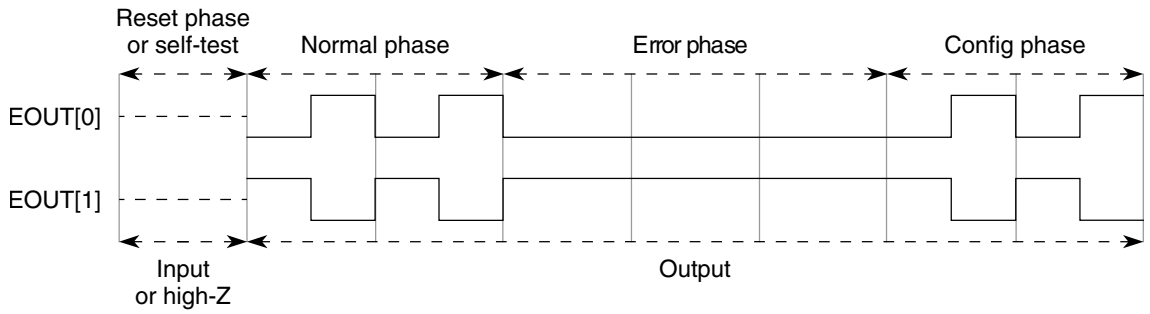


Figure 76-14. Time-switching protocol

76.8.8.8 Bi-stable protocol

The encoding scheme is given in Table 76-8 and the related timing diagram is given in Figure 76-15.

Table 76-8. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

Note

Figure 76-15 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

3. Actual value depends on device specific settings at pad level.

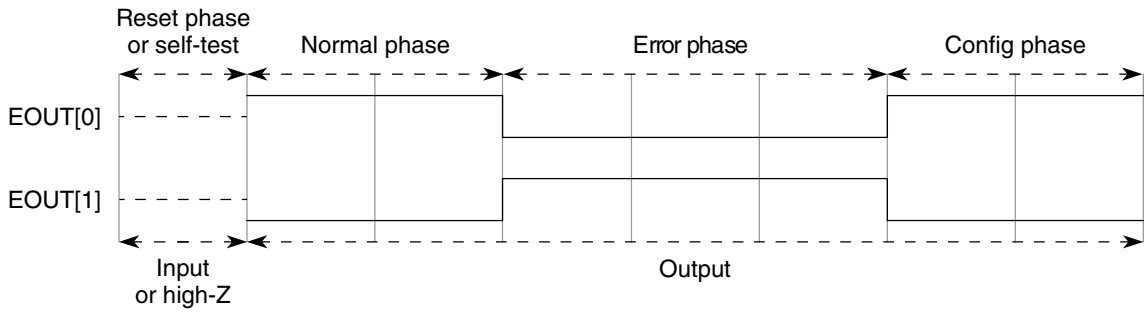


Figure 76-15. Bi-stable protocol

76.8.9 Fault signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

76.9 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don’t read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don’t write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don’t write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible NCF failure source, a different reaction—including no reaction—is configurable through the use of NMI, IRQ, and long/short reset selection registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	76.9.1/3726
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	76.9.2/3728
8	Configuration (FCCU_CFG)	32	R/W	See section	76.9.3/3729
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	76.9.4/3732
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	76.9.4/3732
24	Noncritical Fault Configuration (FCCU_NCF_CFG2)	32	R/W	See section	76.9.4/3732
28	Noncritical Fault Configuration (FCCU_NCF_CFG3)	32	R/W	See section	76.9.4/3732
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	76.9.5/3733
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	76.9.5/3733
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	76.9.5/3733
58	Noncritical Fault State Configuration (FCCU_NCFS_CFG3)	32	R/W	See section	76.9.5/3733
5C	Noncritical Fault State Configuration (FCCU_NCFS_CFG4)	32	R/W	See section	76.9.5/3733
60	Noncritical Fault State Configuration (FCCU_NCFS_CFG5)	32	R/W	See section	76.9.5/3733
64	Noncritical Fault State Configuration (FCCU_NCFS_CFG6)	32	R/W	See section	76.9.5/3733
68	Noncritical Fault State Configuration (FCCU_NCFS_CFG7)	32	R/W	See section	76.9.5/3733
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	76.9.6/3734
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	76.9.6/3734
88	Noncritical Fault Status (FCCU_NCF_S2)	32	R/W	0000_0000h	76.9.6/3734
8C	Noncritical Fault Status (FCCU_NCF_S3)	32	R/W	0000_0000h	76.9.6/3734
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	76.9.7/3736
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	76.9.8/3737
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	76.9.8/3737
9C	Noncritical Fault Enable (FCCU_NCF_E2)	32	R/W	See section	76.9.8/3737
A0	Noncritical Fault Enable (FCCU_NCF_E3)	32	R/W	See section	76.9.8/3737
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	76.9.9/3738
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	76.9.9/3738
AC	Noncritical Fault Timeout Enable (FCCU_NCF_TOE2)	32	R/W	See section	76.9.9/3738
B0	Noncritical Fault Timeout Enable (FCCU_NCF_TOE3)	32	R/W	See section	76.9.9/3738
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	76.9.10/ 3739
B8	Configuration-State Timer Interval (FCCU_CFG_TO)	32	R/W	See section	76.9.11/ 3739

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BC	IO Control (FCCU_EINOUT)	32	R/W	See section	76.9.12/ 3740
C0	Status (FCCU_STAT)	32	R	0000_0010h	76.9.13/ 3742
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	76.9.14/ 3744
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R	0000_0000h	76.9.15/ 3745
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R	0000_0000h	76.9.16/ 3746
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	76.9.17/ 3747
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	76.9.18/ 3748
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	76.9.19/ 3749
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	76.9.20/ 3750
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	76.9.21/ 3751
EC	Mode Controller Status (FCCU_MCS)	32	R	See section	76.9.22/ 3752
F0	Transient Configuration Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	76.9.23/ 3754
F4	Permanent Configuration Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	76.9.24/ 3755
F8	Delta T (FCCU_DELTA_T)	32	R/W	0000_0000h	76.9.25/ 3755
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	76.9.26/ 3756
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	76.9.26/ 3756
104	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	76.9.26/ 3756
108	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN3)	32	R/W	0000_0000h	76.9.26/ 3756
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	76.9.27/ 3757
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	76.9.27/ 3757
114	NMI Enable (FCCU_NMI_EN2)	32	R/W	0000_0000h	76.9.27/ 3757
118	NMI Enable (FCCU_NMI_EN3)	32	R/W	0000_0000h	76.9.27/ 3757

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	76.9.28/ 3758
120	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	76.9.28/ 3758
124	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	76.9.28/ 3758
128	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN3)	32	R/W	0000_0000h	76.9.28/ 3758

76.9.1 Control (FCCU_CTRL)

Initiates and indicates the status of operations, indicates that FCCU successfully loaded configuration data from the NVMCFG signals, and enables Debug mode.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0							0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

FCCU_CTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. **NOTE:** FOSU doesn't halt when FCCU enters Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in Debug mode. 0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation Run Initiates operations that perform actions such as putting FCCU in Configuration state or reading a timer. For information on how to run operations, see Run operations. FCCU ignores any write to this field while the operation status (OPS) is In Progress. After completion of an operation, FCCU sets this field to OP0. The following events result in an operation status (OPS) of Aborted: • Writing to a Noncritical Fault Status (FCCU_NCF_S*n*) register (which automatically initiates the OP12 operation) without first successfully unlocking the register (NCFK) • Initiating an OP1 operation when FCCU is not in Normal state or the configuration is locked • Initiating an OP1, OP2, or OP31 operation without first unlocking the operation (CTRLK) 00000 OP0—No operation 00001 OP1—Applies only when the configuration is unlocked, when FCCU is in Normal state, and immediately after you unlock the operation (CTRLK). Put FCCU in Configuration state. 00010 OP2—Applies only immediately after you unlock the operation (CTRLK). Put FCCU in Normal state. 00011 OP3—Capture FCCU's state and the states of the EOUT signals in the Status (FCCU_STAT) register. 00100 OP4—Read the FCCU frozen status flags (NA Freeze Status (FCCU_N2AF_STATUS)). 00101 OP5—Read the FCCU frozen status flags (AF Freeze Status (FCCU_A2FF_STATUS)). 00110 OP6—Read the FCCU frozen status flags (NF Freeze Status (FCCU_N2FF_STATUS)).

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	00111 OP7—Read the FCCU frozen status flags (FA Freeze Status (FCCU_F2A_STATUS)). 01000 Reserved 01001 Reserved 01010 OP10—Read the Noncritical Fault Status (FCCU_NCF_S*n*) register. 01011 Reserved 01100 OP12—Do not initiate this operation; it is automatically initiated by FCCU. A Noncritical Fault Status (FCCU_NCF_S*n*) register status clear operation is in progress. 01101 OP13—Clear the freeze status registers (see the freeze registers). 01110 OP14—Do not initiate this operation; it is automatically initiated by FCCU. A Configuration-state timeout is in progress. For more information, see Configuration registers. 01111 OP15—Set the operation status (OPS) to Idle. 10000 Reserved 10001 OP17—Read the Alarm-state timer (XTMR). 10010 Reserved 10011 OP19—Read the Configuration-state timer (XTMR). 10100 Reserved 10101–11110 Forbidden. Writing any of these values returns an operation status (OPS) of Aborted with no side effect. 11111 OP31—Applies only immediately after you unlock the operation (CTRLK). Load configuration data from the NVMCFG signals to certain fields in the Configuration (FCCU_CFG) register, but without resetting FCCU (for testing and debugging). See FCCU_CFG register bit value sources (N and C) by event. The operation status (OPS) for this operation is always Successful.

76.9.2 Control Key (FCCU_CTRLK)

Writable only with a 32-bit write. Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation.

NOTE

Reading from this register always returns the value 0000_0000h.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Locked-Operation Control Key Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation. **NOTE:** You must initiate an operation in the instruction that immediately follows the instruction that unlocks it; otherwise, the operation is again locked. 9137_56AFh: Unlock OP1. 825A_132Bh: Unlock OP2. 29AF_8752h: Unlock OP31. Any other value: Do nothing.

76.9.3 Configuration (FCCU_CFG)

Writable only when FCCU is in Configuration state. Specifies the global configuration for FCCU.

The values of the bits in this register are affected by writes as well as any of the following events:

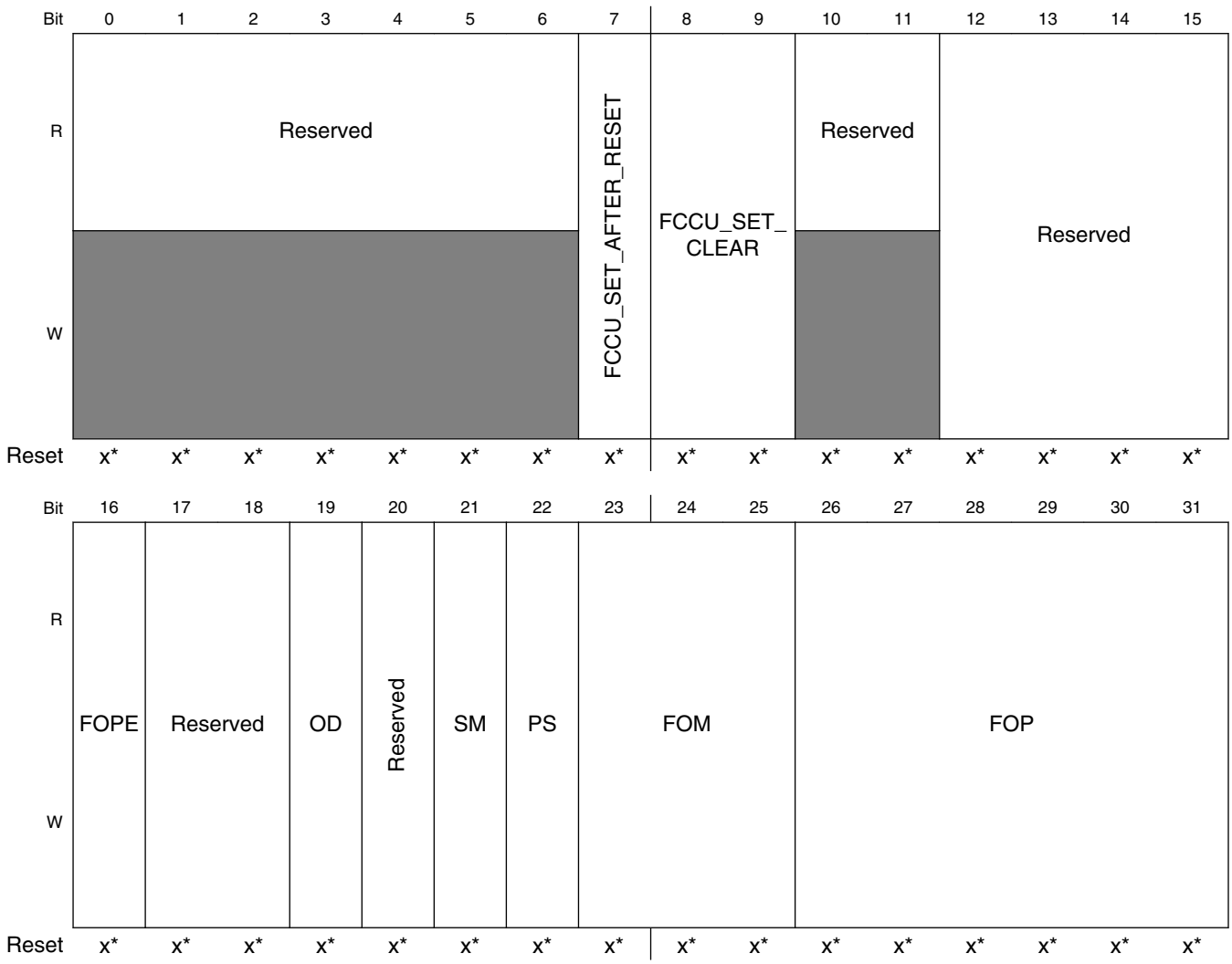
- The chip resets FCCU.
- FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires. This is called a Configuration-state timeout.
- You run an OP31 operation.

For information on the sources of bit values for these events, see FCCU_CFG register bit value sources (N and C) by event.

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the fault-output (EOUT) timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the fault-output (EOUT) timer expires (FCCU stops indicating a fault on the EOUT signals).

Address: 0h base + 8h offset = 8h



* Notes:

- For information on this register's reset value, see FCCU_CFG register bit value sources (N and C) by event.x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 FCCU_SET_AFTER_RESET	Fault-Output (EOUT) Activate For fault-output (EOUT) signaling, controls whether the EOUT signals are active. 0 Inactive (the EOUT signals are in a high-impedance state) 1 Active (the EOUT signals indicate FCCU's condition)
8–9 FCCU_SET_CLEAR	Fault-Output (EOUT) Control Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Controls whether the fault-output (EOUT) signals are controlled by FCCU's finite state machine (FSM).

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	00 Controlled by the FSM 01 Always low 10 Controlled by the FSM 11 High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running.
10–11 Reserved	This field is reserved.
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 FOPE	Fault-Output (EOUT) Prescaler Extension For fault-output (EOUT) signaling, specifies the most significant bit of the derived prescaler that controls the signaling frequency. For more information, see the description of the FOP field.
17–18 Reserved	This field is reserved. Always write the reset value to this field.
19 OD	Open-Drain Mechanism to select between Push-pull and Open drain(OD) mode for the error indicating pin(s) 0 Push-pull 1 Open-drain
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes. 0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. **NOTE:** In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used]

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 FOP	Fault-Output (EOUT) Prescaler For fault-output (EOUT) signaling, specifies the least significant bits of the derived prescaler that controls the signaling frequency. The derived prescaler is the concatenation of FOPE and FOP and is represented by *{FOPE, FOP}*. The following equation defines the signaling frequency in terms of the CLKSAFE frequency ($CLKSAFE_{freq}$) and the derived prescaler: $EOUT_{freq} = CLKSAFE_{freq} / ((\{FOPE, FOP\} + 1) \times 2 \times 2048)$ The following example values and meanings are for the derived (7-bit) prescaler, not FOP alone. The meanings show the derived prescaler followed by the corresponding value by which the input clock frequency is then divided. 0000000: 0; divide by 2 × 2048 (4096) 0000001: 1; divide by 4 × 2048 (8192) 0000010: 2; divide by 6 × 2048 (12,288) … 0001000: 8; divide by 18 × 2048 (36,864) … 1111101: 125; divide by 252 × 2048 (516,096) 1111110: 126; divide by 254 × 2048 (520,192) 1111111: 127; divide by 256 × 2048 (524,288)

76.9.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

These registers are writable only when FCCU is in Configuration state.

Noncritical Fault Configuration (FCCU_NCF_CFG*n*) register (value of n)	Offset	NCFCx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	1Ch	31	0
1	20h	63	32
2	24h	95	64
3	28h	127	96

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFG*n* field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag. For the mapping of the NCFCx fields among registers, see the earlier table in this register description. 0 Hardware-recoverable fault 1 Software-recoverable fault

76.9.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)

The FCCU_NCFS_CFGx registers contain the configuration of each noncritical fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 76-9 shows FCCU configuration register channels.

Table 76-9. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]
58h	FCCU_NCFS_CFG3	NCFSCx[63:48]
5Ch	FCCU_NCFS_CFG4	NCFSCx[79:64]
60h	FCCU_NCFS_CFG5	NCFSCx[95:80]
64h	FCCU_NCFS_CFG6	NCFSCx[111:96]
68h	FCCU_NCFS_CFG7	NCFSCx[127:112]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFG*n* field descriptions

Field	Description
0–31 NCFSCx	Noncritical fault state configuration See Table 76-9 for register offset to channel number relationship. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

76.9.6 Noncritical Fault Status (FCCU_NCF_S*n*)

The FCCU_NCF_Sx register contains the latched fault indication collected from the noncritical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Noncritical Fault Status (FCCU_NCF_S*n*) register (value of n)	Offset	NCFSx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	80h	31	0
1	84h	63	32
2	88h	95	64
3	8Ch	127	96

Address: 0h base + 80h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_S*n* field descriptions

Field	Description
0–31 NCFSx	Noncritical fault status. The status bits related to the noncritical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. For the mapping of the NCFSx fields among registers, see the earlier table in this register description. 0 No "noncritical" fault latched 1 "Noncritical" fault latched

76.9.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Noncritical fault key = AB34_98FEh

76.9.8 Noncritical Fault Enable (FCCU_NCF_E*n*)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Noncritical Fault Enable (FCCU_NCF_E*n*) register (value of n)	Offset	NCFEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	94h	31	0
1	98h	63	32
2	9Ch	95	64
3	A0h	127	96

Address: 0h base + 94h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_E*n* field descriptions

Field	Description
0–31 NCFEx	Noncritical fault enable For the mapping of the NCFEx fields among registers, see the earlier table in this register description. 0 No actions following the respective noncritical fault assertion 1 FCCU moves to ALARM or FAULT state

76.9.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)

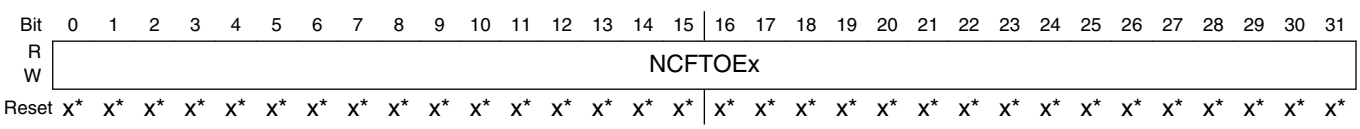
The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective noncritical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the noncritical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related noncritical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) register (value of n)	Offset	NCFTOEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	A4h	31	0
1	A8h	63	32
2	ACh	95	64
3	B0h	127	96

Address: 0h base + A4h offset + (4d × i), where i=0d to 3d



* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOE*n* field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable For the mapping of the NCFTOEx fields among registers, see the earlier table in this register description. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

76.9.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state. The alarm timeout value should be programmed to be less than FOSU_COUNT, or destructive resets may be generated by FOSU timeouts.

NOTE

This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × $T_{RC16MHz}$

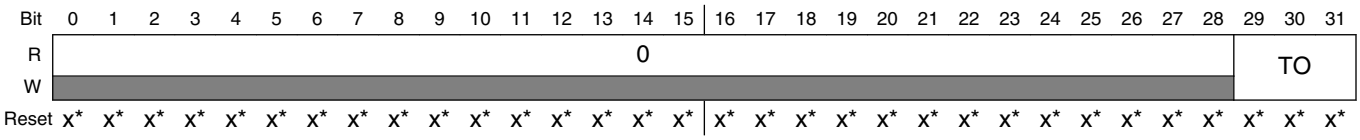
76.9.11 Configuration-State Timer Interval (FCCU_CFG_TO)

Writable only when FCCU is in Normal, Alarm, or Fault state (not in Configuration state). Not accessible while a Configuration-state timeout (OP14 operation) is in progress. Controls the maximum amount of time that FCCU can be in Configuration state.

NOTE

When a Configuration-state timeout occurs, FCCU changes the value of this register to its reset value.

Address: 0h base + B8h offset = B8h



* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration-State Timer Interval Controls the maximum amount of time (T_max) that FCCU can be in Configuration state according to this equation: $T\_max = T_{16MHz} \times 2^{(TO + 10)}$ If you put FCCU in Configuration state and this timer interval expires (called a Configuration-state timeout), then FCCU: • Automatically leaves Configuration state and enters Normal state. • Changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration (FCCU_CFG) register's Configuration-state-timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers. **NOTE:** Make sure the Configuration-state timer interval is less than the FOSU module's timer interval; otherwise, a fault that occurs while FCCU is in Configuration state can cause FOSU to generate a chip reset. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information. 000: 64 µs ... 111: 8192 µs

76.9.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)

- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 76-10 shows Bi-stable encoding.

Table 76-10. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										EIN1	EIN0	0		EOUT1	EOUT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	x*	x*	0	0	0	0

* Notes:
- x = Undefined at reset.

FCCU_EINOUT field descriptions

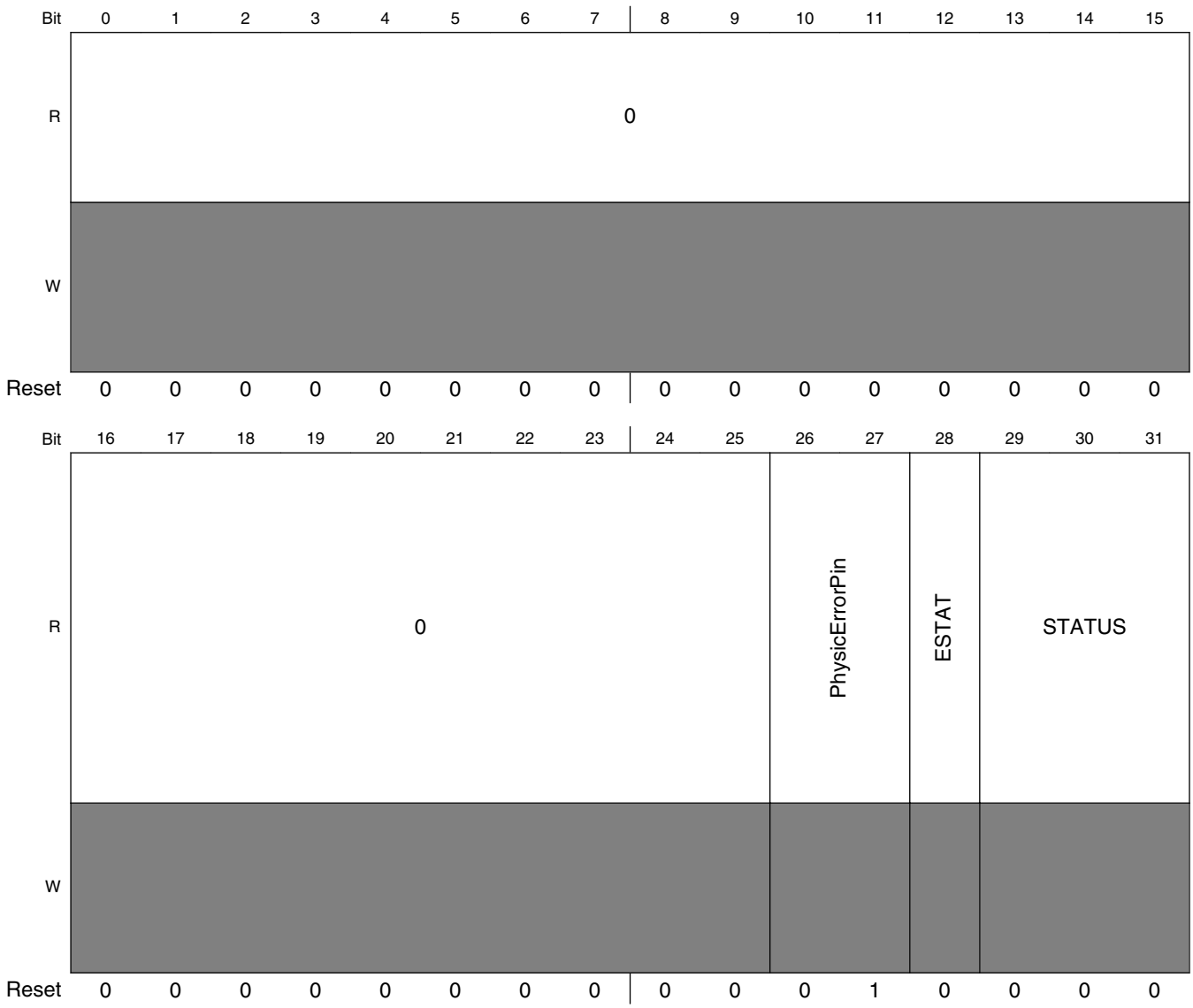
Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error Input 1 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN1 signal. 0 Low 1 High
27 EIN0	Error Input 0 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN0 signal. 0 Low 1 High
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1
31 EOUT0	Error out 0 (significative only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

76.9.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h



FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	Fault Output (EOUT) States Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals. 00 EOUT1 is low; EOUT0 is low. 01 EOUT1 is low; EOUT0 is high. 10 EOUT1 is high; EOUT0 is low. 11 EOUT1 is high; EOUT0 is high.
28 ESTAT	Fault State

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
	Applies only after successfully running an OP3 operation (OPR). Indicates whether FCCU is in Fault state. 0 Not in Fault state (in Normal, Alarm, or Configuration state) 1 In Fault state
29–31 STATUS	FCCU State Applies only after successfully running an OP3 operation (OPR). Indicates FCCU's state. 000 Normal 001 Configuration 010 Alarm 011 Fault 100 Reserved 101 Reserved 110 Reserved 111 Reserved

76.9.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "noncritical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

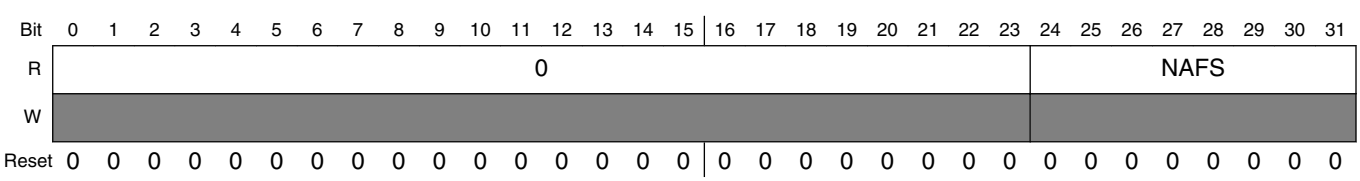
The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h



FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

76.9.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (noncritical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						AF_ SRC		AFFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple and/or "noncritical" faults
24–31 AFFS	Alarm to Fault Frozen Status 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault ... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

76.9.16 NF Freeze Status (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the noncritical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh

Bit	0–15	16–21	22–23	24–31
R	0	0	NF_SRC	NFFS
W				
Reset	0	0	0	0

FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple "noncritical" faults
24–31 NFFS	Normal to Fault Frozen Status 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

76.9.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the noncritical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

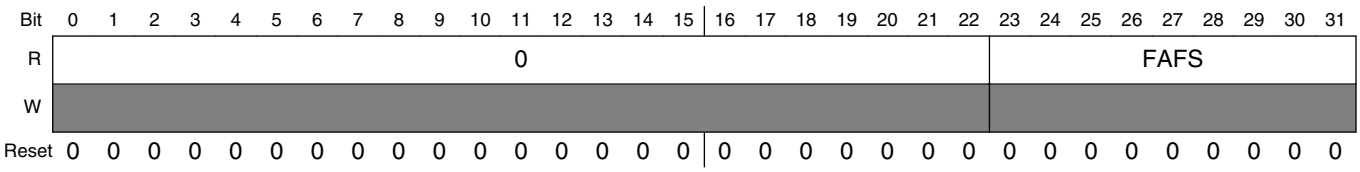
1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.

2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h



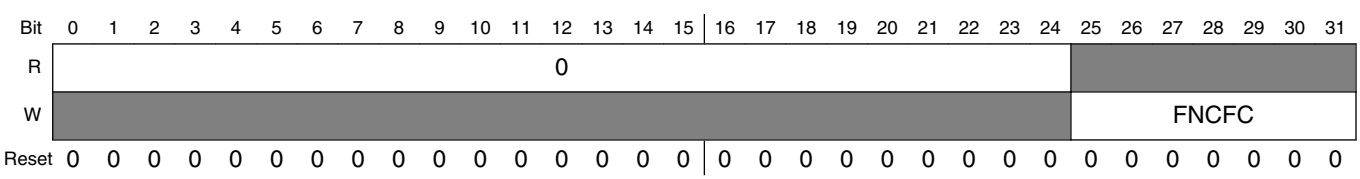
FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

76.9.18 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a noncritical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the noncritical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake noncritical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each noncritical fault injection.

Address: 0h base + DCh offset = DCh



FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake noncritical fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake noncriticalfault injection at noncriticalfault source 0 01h: Fake noncriticalfault injection at noncriticalfault source 1 02h: Fake noncriticalfault injection at noncriticalfault source 2 ... 7Fh: Fake noncriticalfault injection at noncriticalfault source 127

76.9.19 IRQ Status (FCCU_IRQ_STAT)

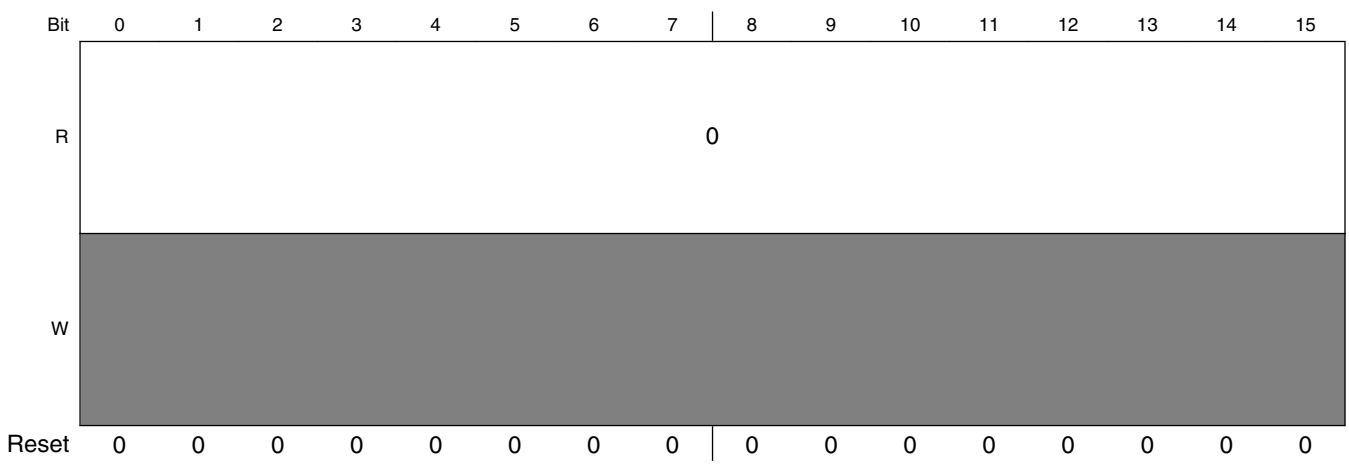
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

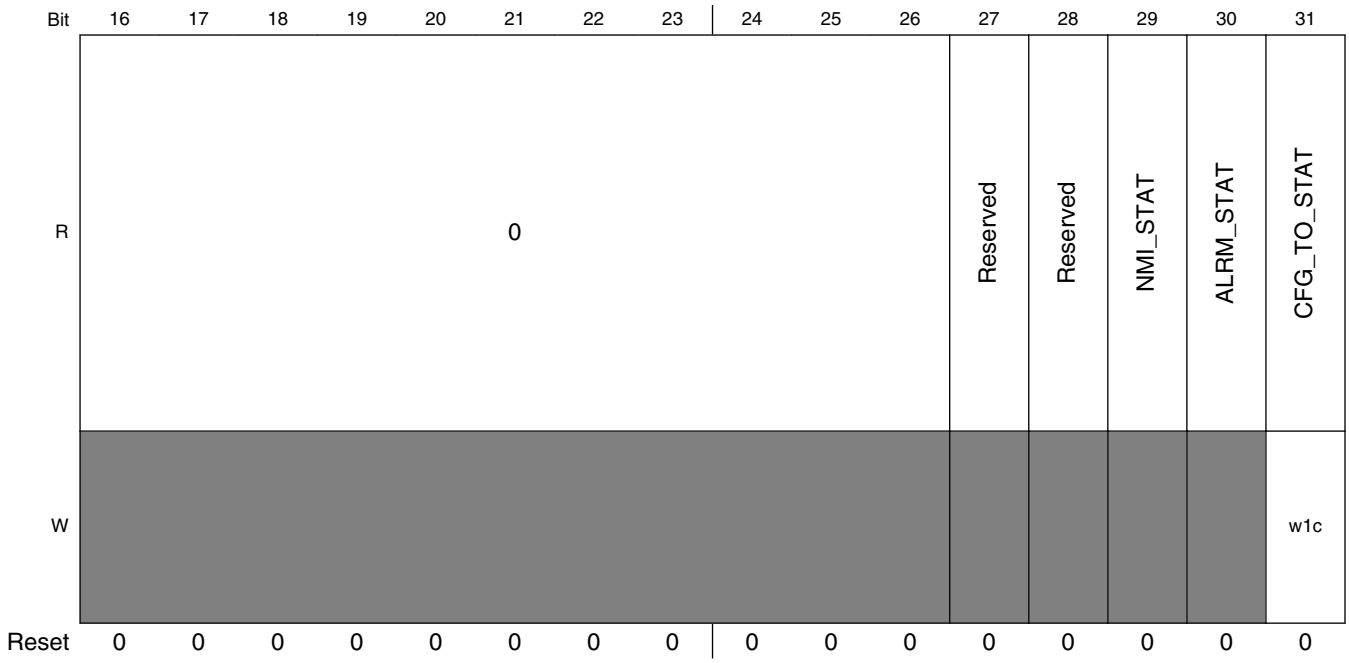
- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h





FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

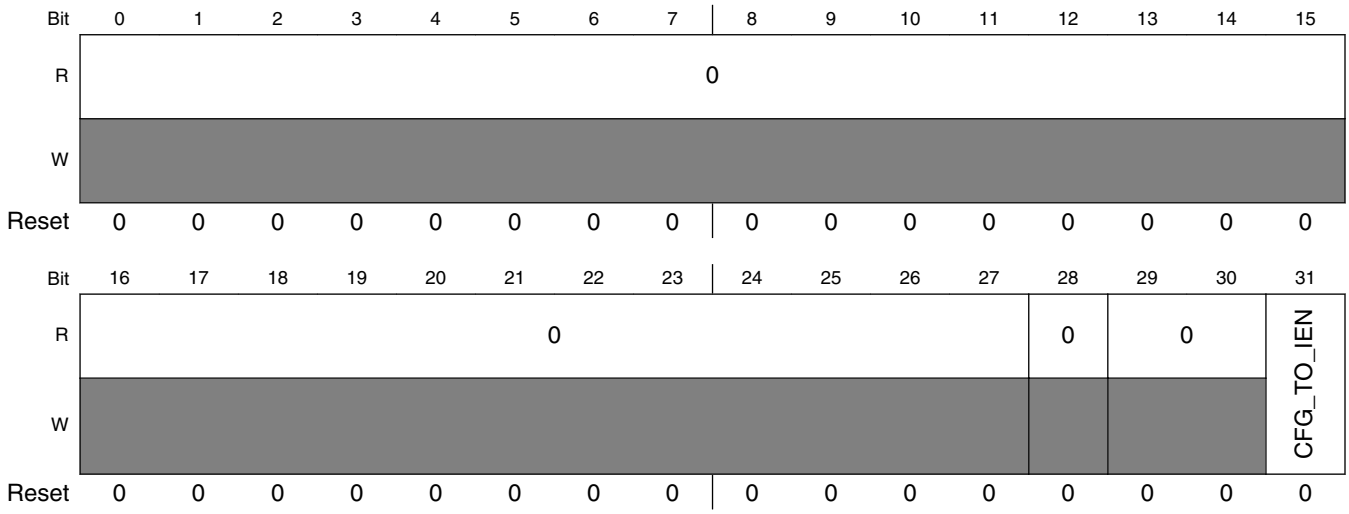
76.9.20 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h



FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

76.9.21 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

The SW application executes the timer read operation by the following sequence:

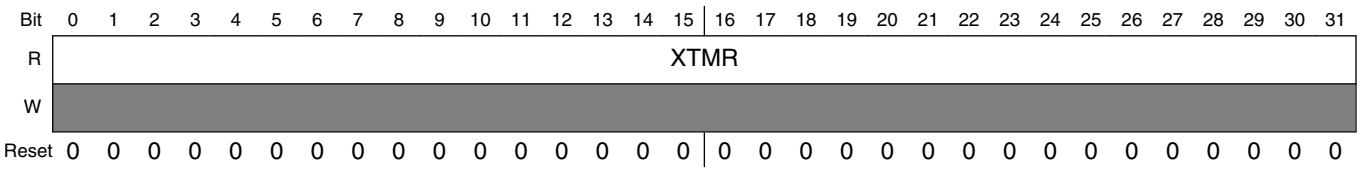
1. Set any of the following operations into the FCCU_CTRL.OPR field:
   - OP17
   - OP19
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 76-11 shows Timer state/value.

Table 76-11. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Address: 0h base + E8h offset = E8h



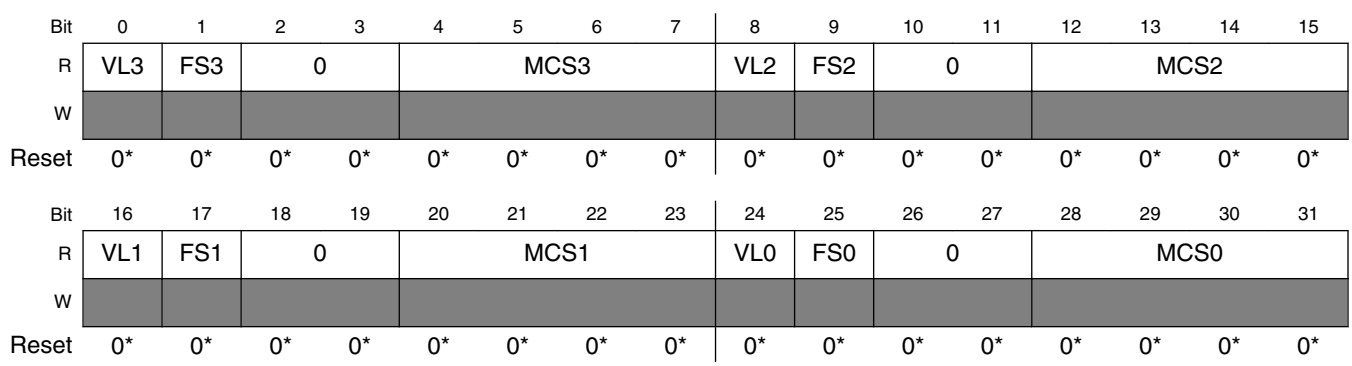
FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

76.9.22 Mode Controller Status (FCCU_MCS)

Indicates the last four chip modes—as defined by the MC_ME module—and whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured each mode.

Address: 0h base + ECh offset = ECh



* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information. (The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.)

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid 3—Indicates whether MCS3 and FS3 are valid. 0 Invalid 1 Valid
1 FS3	Fault Status 3 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS3. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	Mode Controller State 3—Indicates the fourth most recent chip mode.
8 VL2	Valid 2—Indicates whether MCS2 and FS2 are valid. 0 Invalid 1 Valid
9 FS2	Fault Status 2 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS2. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 MCS2	Mode Controller State 2—Indicates the third most recent chip mode.
16 VL1	Valid 1—Indicates whether MCS1 and FS1 are valid. 0 Invalid 1 Valid
17 FS1	Fault Status 1 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS1. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted

Table continues on the next page...

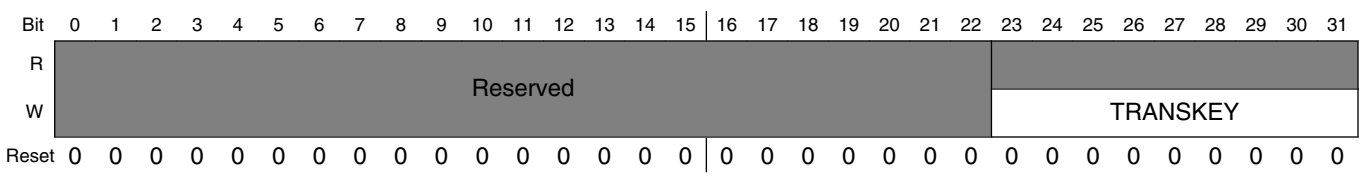
FCCU_MCS field descriptions (continued)

Field	Description
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 MCS1	Mode Controller State 1—Indicates the second most recent chip mode.
24 VL0	Valid 0—Indicates whether MCS0 and FS0 are valid. 0 Invalid 1 Valid
25 FS0	Fault Status 0 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS0. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	Mode Controller State 0—Indicates the most recent (current) chip mode.

76.9.23 Transient Configuration Lock (FCCU_TRANS_LOCK)

Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.

Address: 0h base + F0h offset = F0h



FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 TRANSKEY	Transient Configuration Lock Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state. BCh: Unlock.

Table continues on the next page...

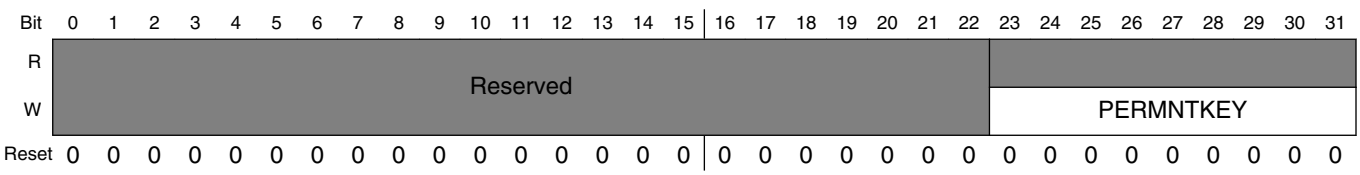
FCCU_TRANS_LOCK field descriptions (continued)

Field	Description
	Any other value: Lock.

76.9.24 Permanent Configuration Lock (FCCU_PERMNT_LOCK)

Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.

Address: 0h base + F4h offset = F4h



FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 PERMNTKEY	Permanent Configuration Lock Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset. FFh: Lock. Any other value: Do nothing.

76.9.25 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Bistable Minimum Fault-Output (EOUT) Faulty Interval Applies only to Bistable fault-output mode (FOM). Controls the minimum amount of time (T_min) that the fault-output (EOUT) timer runs according to this equation: T_min = 250 µs + DELTA_T The following values and meanings are for DELTA_T: 00_0000_0000_0000: 0 µs 00_0000_0000_0001: 1 µs … 11_1111_1111_1110: 16,382 µs 11_1111_1111_1111: 16,383 µs **NOTE:** The durations shown for the DELTA_T values are for CLKSAFE clock signals running at exactly 16 MHz. However, the CLKSAFE signals are sourced from an internal chip clock signal (for example, IRC)—or an integer division of it (see the chip-specific FCCU information)—that has a trimmed frequency variation. Therefore, the DELTA_T durations may vary. See your chip's data sheet for the trimmed frequency variation (for example, δF_{var}).

76.9.26 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*) register (value of n)	Offset	IRQENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	0FCh	31	0
1	100h	63	32
2	104h	95	64
3	108h	127	96

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable For the mapping of the IRQENx fields among registers, see the earlier table in this register description. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

76.9.27 NMI Enable (FCCU_NMI_EN*n*)

These registers enable the NMI.

Table 76-12 shows FCCU NMI enable register channels.

Table 76-12. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]
118h	FCCU_NMI_EN3	NMIENE[127:96]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 76-12 for register offset to channel number relationship. 0 NMI is disabled for error (NCF) source x. 1 NMI is enabled for error (NCF) source x.

76.9.28 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.)

When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.)

Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register (value of n)	Offset	EOUTENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	11Ch	31	0
1	120h	63	32
2	124h	95	64
3	128h	127	96

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	Bistable Fault-Output (EOUT) Mode Signaling Enable x Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.) When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.) **NOTE:** When FCCU is configured for a fault-output mode other than Bistable, you must set this field to enabled to ensure that FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x) so the FOSU module doesn't mistakenly generate a destructive chip reset. For the mapping of the EOUTENx fields among registers, see the earlier table in this register description. 0 Disabled 1 Enabled

76.9.29 Configuration registers

76.9.29.1 Definition

Configuration registers are registers that:

- Let you configure FCCU's Alarm-state timer interval, fault channels, and fault-output (EOUT) signals.
- You can write to only when the configuration is not locked (see Lock and unlock the configuration) and FCCU is in Configuration state (see Put FCCU in Configuration or Normal state).
- Save the values you write to them while FCCU is in Configuration state only after you manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the FCCU_CFG register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event.

76.9.29.2 The configuration registers

The following registers are the configuration registers, listed in offset order from lowest to highest:

- Configuration (FCCU_CFG)
- Noncritical Fault Configuration (FCCU_NCF_CFG*n*)
- Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)
- Noncritical Fault Enable (FCCU_NCF_E*n*)
- Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)
- Noncritical Fault Timeout (FCCU_NCF_TO)
- Delta T (FCCU_DELTA_T)
- IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)
- NMI Enable (FCCU_NMI_EN*n*)
- Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

76.9.30 FCCU_CFG register bit value sources (N and C) by event

Certain events affect the bit values of the Configuration (FCCU_CFG) register; when one of these events occurs, the bit value is either unaffected, or FCCU changes it to one of the following:

- N: a chip-specific (or, for some chips, user-specified) value provided by the bit's associated NVMCFG signal
- C: a chip-specific value provided by the bit's associated constant inside the module

For each bit in the Configuration (FCCU_CFG) register, this table shows the associated NVMCFG signal, if any, and indicates the source of the bit value for the events that affect it. For specific values of *N* and *C* for each bit, see the chip-specific FCCU information.

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
0	21	N	C	N
1	20	N	C	N
2	19	N	C	N
3	18	N	C	N
4	17	N	C	N
5	16	N	C	N
6	15	N	C	N
7	—	C	C	(Unaffected)

Table continues on the next page...

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
8	—	C	C	(Unaffected)
9	—	C	C	(Unaffected)
10	—	C	C	(Unaffected)
11	—	C	C	(Unaffected)
12	—	C	C	(Unaffected)
13	—	C	C	(Unaffected)
14	—	C	C	(Unaffected)
15	—	C	C	(Unaffected)
16	14	N	C	N
17	13	N	C	N
18	12	N	C	N
19	—	C	C	(Unaffected)
20	11	N	C	N
21	10	N	C	N
22	9	N	C	N
23	8	N	C	N
24	7	N	C	N
25	6	N	C	N
26	5	N	C	N
27	4	N	C	N
28	3	N	C	N
29	2	N	C	N
30	1	N	C	N
31	0	N	C	N

76.10 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with

the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., a disabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU, which is asserted on power-on, 'destructive', and external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction.

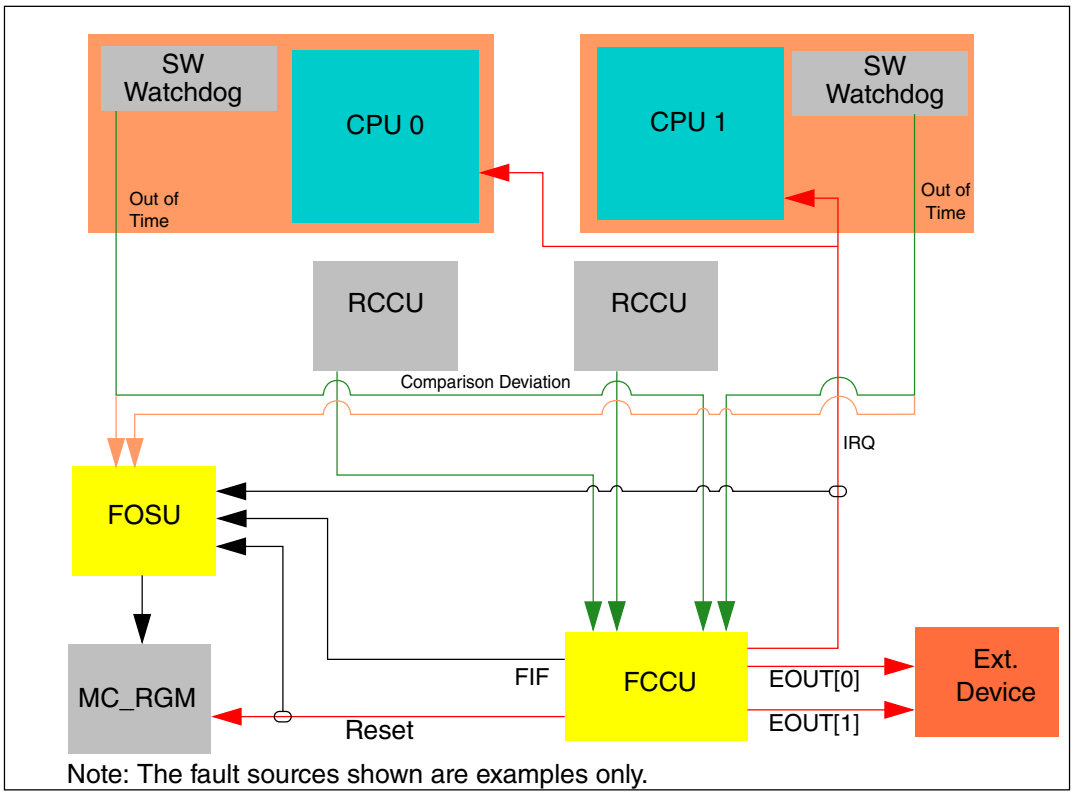


Figure 76-16. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

76.11 Use cases and limitations

76.11.1 Configuration guidelines

Follow these guidelines to configure FCCU:

- If you want FCCU to react to a fault on a noncritical fault channel:
  - Enable the channel (NCFEx).
  - Enable at least one type of Fault-state reaction for the channel: chip reset (NCFSCx), fault-output (EOUT) signaling (EOUTENx), or nonmaskable interrupt (NMIENx).
  - If you enable chip reset as the type of Fault-state reaction for the channel (NCFSCx), enable either Alarm state (NCFTOEx) or at least one other type of Fault-state reaction for the channel: fault-output (EOUT) signaling (EOUTENx) or nonmaskable interrupt (NMIENx).
  - If you enable Alarm state for the channel (NCFTOEx), enable the Alarm-state reaction (IRQENx).
  - If you enable Alarm state for the channel (NCFTOEx), make sure the Alarm-state timer interval (TO) is less than the FOSU module's timer interval; otherwise, FOSU generates a chip reset every time a fault occurs on the channel. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information.

NOTE

If you enable a noncritical fault channel but disable all reactions for that channel, FCCU changes state if necessary but doesn't perform any reaction because reactions are disabled. If you enable reactions for a noncritical fault channel but disable that channel, and FCCU is in Normal state when a fault occurs on the channel, FCCU doesn't enter Alarm or Fault state and therefore doesn't perform any reaction.

76.11.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out T_min to expire
   b. Check and clear fault status
   c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires

d. Verify FCCU is in NORMAL state, else repeat step(a) above
e. Read and verify value in FCCU_NCF_Ex
f. Reconfigure FCCU, if necessary

Chapter 77
Self-Test Control Unit (STCU2)

77.1 Chip-specific STCU2 information

77.1.1 STCU2 configuration

The STCU includes eight LBIST partitions and 92 MBIST CUTs.

The Self-Test can be initiated as follows:

- Offline Self-Test can be initiated whenever a Destructive/Long external reset/POR reset occurs
- Online Self-Test can be initiated whenever the Software Programs

Before starting self-test, make sure that:

- All previous applications are terminated
- PLL1 should be configured for 240 MHz PHI clock
- PLL1 should be selected as the system clock
- PLL1 should be selected as source for all auxiliary clocks

NOTE

If an erase (or program) operation is going on in the Flash, then any request for LBIST, or any customer accessible self test mechanism that utilizes the flash scan interface should be disabled. Ensure Flash is in IDLE state before starting LBIST.

For running online self-test, choose maximum STCU2_CFG[LB_DELAY] and the maximum shift frequency is 45 MHz for 4:4:2:1 clock mode and 60 MHz for 4:2:2:1 clock mode or 2:2:2:1 clock mode. Shut-down self-test is a self-test configured and initiated by software (online self-test). Since the self-test is destructive, it is supposed to be run at shut-down if desired at all.

NOTE

Pattern Count and MISR values are dependent on test coverage requirements for the system.

NOTE

An HVD event on VDD_LV_CORE or VDD_HV_ADC will not prevent the offline STCU self-test from executing if it is enabled and the HVD is masked through DCF configuration. If the HVD remains engaged when the STCU completes the self-test, a destructive reset will occur causing the offline STCU self-test to repeat continually until the HVD condition is removed.

The following table describes the mapping between the MBIST number, as the STCU2 captures it, and the appropriate memory.

NOTE

DTCM in the table below is referred to as "DMEM" in the Core Complex Overview and the Core description chapters.

MBIST partitions are referred as "NMCUTs" in the STCU chapter.

Table 77-1. MBIST mapping

Partition number	Address Range (If applicable)	Memory
0	0xFFFFC000 - 0xFFFFFFFF	BAM_ROM
1	Memory Mapping mentioned in Section Error Injection Address Register (CAN_ERRIAR)	CAN_0
2	Memory Mapping mentioned in Section CAN_1 Error Injection Address Register (CAN_ERRIAR) - error injection addresses	CAN_1
3	IP-Internal / Not System Memory Mapped	FLEXRAY DATA
4	Following Registers are mapped to LUT RAM: "Message Buffer Cycle Counter Filter Register (FR_MBCCFRn), Message Buffer Frame ID Register (FR_MBFIDRn), Message Buffer Index Register (FR_MBIDXRn), Message Buffer Data Field Offset Register (FR_MBDORn), LRAM ECC Error Test Register (FR_LEETRn)"	FLEXRAY LUT
5	IP-Internal / Not System Memory Mapped	ETHERNET TX
6	IP-Internal / Not System Memory Mapped	ETHERNET RX
7	IP-Internal / Not System Memory Mapped	CORE0 DCACHE

Table continues on the next page...

Table 77-1. MBIST mapping (continued)

Partition number	Address Range (If applicable)	Memory
8		
9		
10		
11	IP-Internal / Not System Memory Mapped	CORE0 DTAG
12	0x50808000 - 0x5080FFFF	CORE0 DTCM
13	0x50800000 - 0x50807FFF	
14	IP-Internal / Not System Memory Mapped	CORE0 ICACHE
15		
16		
17		
18	IP-Internal / Not System Memory Mapped	CORE0 ITAG
19	IP-Internal / Not System Memory Mapped	CORE1 DCACHE
20		
21	IP-Internal / Not System Memory Mapped	CORE1 DTAG
22	0x50818000 - 0x5081FFFF	CORE1 DTCM
23	0x50810000 - 0x50817FFF	
24	IP-Internal / Not System Memory Mapped	CORE1 ICACHE
25		
26	IP-Internal / Not System Memory Mapped	CORE1 ITAG
27	IP-Internal / Not System Memory Mapped	CORE2 DCACHE
28		
29	IP-Internal / Not System Memory Mapped	CORE2 DTAG
30	0x50828000 - 0x5082FFFF	CORE2 DTCM
31	0x50820000 - 0x50827FFF	
32	IP-Internal / Not System Memory Mapped	CORE2 ICACHE
33		
34	IP-Internal / Not System Memory Mapped	CORE2 ITAG
35	0xFC0A1000 - 0xFC0A1400	DMA
36	0x40010000 - 0x4001FFFF	PRAM0
37	0x40000000 - 0x4000FFFF	
38	0x40030000 - 0x4003FFFF	PRAM1
39	0x40020000 - 0x4002FFFF	
40	0x40050000 - 0x4005FFFF	PRAM2
41	0x40040000 - 0x4004FFFF	
42	0x40070000 - 0x4007FFFF	PRAM3

Table continues on the next page...

Table 77-1. MBIST mapping (continued)

Partition number	Address Range (If applicable)	Memory
43	0x40060000 - 0x4006FFFF	
44	0x40090000 - 0x4009FFFF	PRAM4
45	0x40080000 - 0x4008FFFF	
46	0x400B0000 - 0x400BFFFF	
47	0x400A0000 - 0x400AFFFF	
48	0x400D0000 - 0x400DFFFF	PRAM5
49	0x400C0000 - 0x400CFFFF	
50	0x400F0000 - 0x400FFFFF	
51	0x400E0000 - 0x400EFFFF	
52	IP-Internal / Not System Memory Mapped	Reserved
53	IP-Internal / Not System Memory Mapped	
54	IP-Internal / Not System Memory Mapped	SPT
55		
56		
57		
58		
59		
60		
61		
62		
63		
64		
65		
66		
67		
68		
69		
70	IP-Internal / Not System Memory Mapped	SPT TWIDDLE
71		
72		
73		
74		
75		
76		
77		
78	IP-Internal / Not System Memory Mapped	CWG LUT
79		
80		

Table continues on the next page...

Table 77-1. MBIST mapping (continued)

Partition number	Address Range (If applicable)	Memory
81		
82	IP-Internal / Not System Memory Mapped	Reserved
83	Memory Mapping mentioned in Section Error Injection Address Register (CAN_ERRIAR)	CAN_2
84	0x40110000 - 0x4011FFFF	PRAM6
85	0x40100000 - 0x4010FFFF	
86	0x40130000 - 0x4013FFFF	
87	0x40120000 - 0x4012FFFF	
88	0x40150000 - 0x4015FFFF	PRAM7
89	0x40140000 - 0x4014FFFF	
90	0x40170000 - 0x4017FFFF	
91	0x40160000 - 0x4016FFFF	

Table 77-2. LBIST mapping

LBIST partition	Partition Index
peri_a_intc (INTC, ENET)	0
peri_b (AIPS1, PBRIDGE_1 peripherals)	1
chk_z4z7b (RCCU,Checker core)	2
comp_core_sh (Core0, Core1)	3
comp_peri_sh (MEMU,Flash, PRAM_CTRL, PFLASH_CTRL)	4
peri_a_rest (AIPS0, PBRIDGE_0 peripherals)	5
Spt (SPT, WGM,CSI,CTE)	6
debug_ana (MC_ME, JTAG)	7

77.1.1.1 Supported BIST sequences

NXP has validated and therefore supports only particular STCU2 built-in self-test (BIST) sequences for this chip. For descriptions of the supported BIST sequences for this chip, see the engineering bulletin EB00853 (for online self-test) and EB834 (for offline self-test). Contact your NXP sales representative for details.

77.1.1.2 STCU2 register reset values

The following table shows chip-specific reset values for several STCU2 registers. All other register reset values are shown in Register description.

NOTE

These are the reset values when startup self-test is not configured by DCFs. Otherwise, these values change per the DCF configurations of startup self-test.

Table 77-3. STCU2 register reset values

Register	Reset value
STCU2 Configuration Register (STCU2_CFG)	7F00_0000h
STCU2 PLL Configuration Register (STCU2_PLL_CFG)	0000_0000h
STCU2 Watchdog Register Granularity (STCU2_WDG)	0000_FFFFh
STCU2 CRC Expected Status Register (STCU2_CRCE)	FFFF_FFFFh
STCU2 CRC Read Status Register (STCU2_CRCR)	0000_0000h[1]
STCU2 Error FM Register (STCU2_ERR_FM)	0000_0000h
STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)	0000_0000h
STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	0000_0000h
STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	0000_0000h
STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	0000_0000h
STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	0000_0000h
STCU2 LBIST Control Register (STCU2_LB_CTRLn)	0000_0000h
STCU2 LBIST PC Stop Register (STCU2_LB_PCSn)	0000_0000h
STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELn)	FFFF_FFFFh
STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREHn)	FFFF_FFFFh
STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLn)	0000_0000h[1]
STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRHn)	0000_0000h[1]
STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSWn)	FFFF_FFFFh
STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSWn)	FFFF_FFFFh
STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSWn)	0000_0000h[1]
STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSWn)	0000_0000h[1]
STCU2 MBIST Control Register (STCU2_MB_CTRLn)	0000_0000h

1. The value of this register gets updated if offline self-test or online self-test is run.

77.1.1.3 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion

Register write-access watchdog timer refers to two clock-cycle values related to the STCU2's write-access watchdog. The following table provides these values on this chip.

Clock-cycle value	Number of STCU2 clock cycles
AUTOLOCK_VALUE	4096
DCF_COMPLETION	8

77.1.1.4 STCU hardware Abort behavior

If STCU2_ERR_STAT[ABORTHW] bit is set, it is recommended to clear all faults of FCCU.

77.1.1.5 Wait time for writing to the online registers

When writing to the online registers, the user must wait for the expiration of the offline key. The time the user should wait is (IPS_CLK)*4096 clock periods. If the IPS_CLK is 60 MHz then the user must wait (1/60)*4096 = 68.26 μs.

77.1.1.6 On-Line reset generation (only MBIST)

The On-Line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences related to the write operations performed by the MBIST execution other than preventing any read/write operation during the Self-Test execution. If necessary, the software must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

77.1.1.7 On-Line reset generation (LBIST enabled)

The LBRMSW register should be configured to generate a global functional reset after LBIST online self-test. It is not recommended to use a dedicated reset due to random data injected by the LBIST Controller.

NOTE

If an ABORT condition is detected by STCU2 during the On-Line Self-Test execution, independently of the value of the LBRMSW register, only the related reset line is pulsed to prevent any potential dangerous system level operations.

77.1.1.8 PLL Loss of Lock during selftest

If PLL-LOL happens when MBIST run is in progress or when LBIST run is in progress, then the selftest run will be aborted with appropriate flags set in STCU2_ERR_STAT register

If the PLL-LOL happens during the small window between the MBIST and LBIST runs, then system moves to safe clock, i.e IRC. There are two possibilities after this:

1. If the LOL event happens before the STCU state machine hits the “CHECK_PLL” state, then STCU waits for the PLL to get locked again.
   - If PLL re-lock happens before timeout window, then the STCU state-machine proceeds as normal. LBIST will be run in the remaining time window till STCU WDOG expires. Depending on the time required to complete LBIST, the LBIST will either complete or time-out.
   - If the PLL re-lock does not happen and the timeout window expires, then STCU exits self-test with time-out reset.
2. If the LOL event happens after the “CHECK_PLL” state, then the LBIST run continues on IRC clock and completes.

77.2 Introduction

The *STCU2 self-test control unit* is a hardware module that:

- Manages the execution of built-in self-tests (BISTs)
- Indicates whether each BIST passed

77.3 Main features

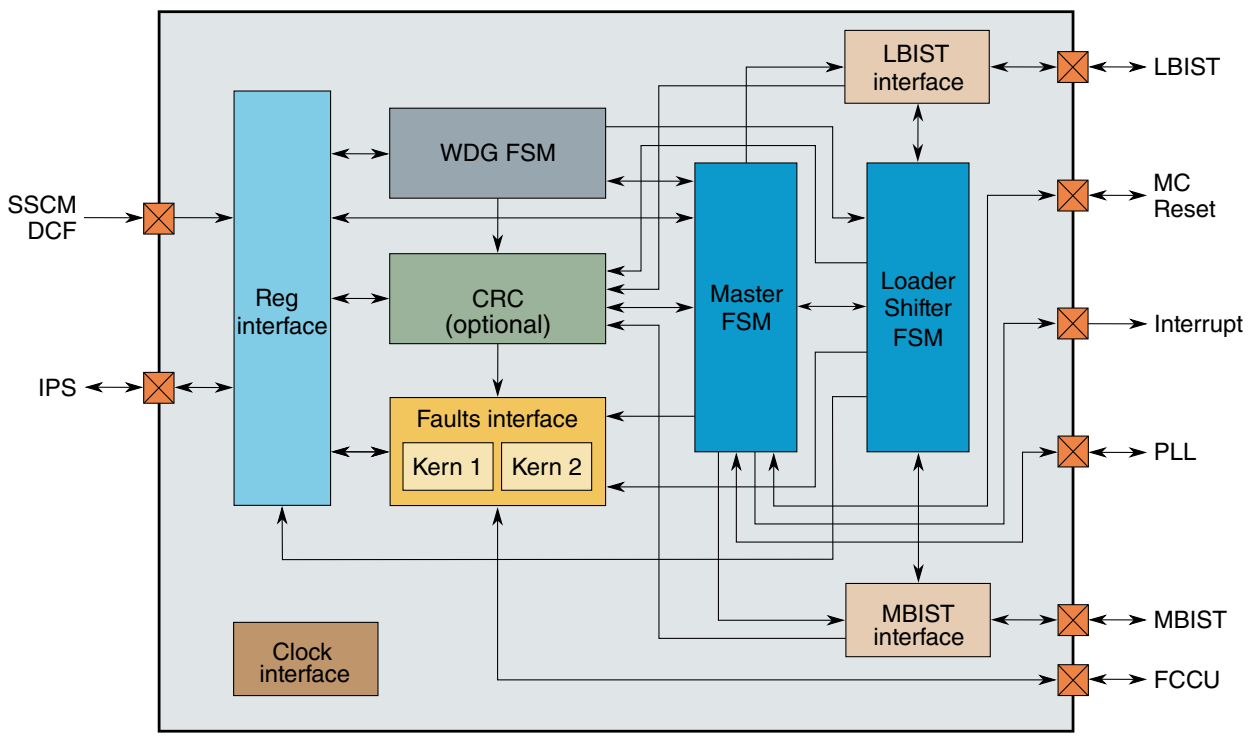
- Internal Peripheral System (IPS) interface for reading from and writing to the STCU2 registers during runtime (online) via the CPU
- SSCM Device Configuration Format (DCF) interface for writing to the STCU2 registers from nonvolatile memory during reset (offline) via the SSCM module (has priority over the IPS interface)
- Programmable scheduler for BIST execution
- LBIST concurrent/sequential execution
- MBIST concurrent/sequential execution
- Programmable LBIST delayed concurrent start
- 32-bit CRC for STCU2 self-checking capability (enabled by CRCEN)
- Programmable internal clock prescaler to reduce internal and BIST TCK clocks
- Programmable PLL direct control during offline BIST sequence
- PLL Lock signal monitoring during BIST sequences

- Programmable BIST-sequence-execution watchdog timer that specifies the maximum time allowed for the execution of a BIST sequence
- Fixed offline-initialization watchdog timer that specifies the maximum time allowed for offline initialization before indicating an initialization fault
- Fixed register-write-access watchdog timer that specifies the maximum amount of time that STCU2 allows you to write to its registers after you unlock them
- Fields that indicate the execution status of each:
  - Online MBIST
  - Offline MBIST
  - Online LBIST
  - Offline LBIST
- Fields that indicate the status of each type of online STCU2 internal error condition
- Fields that indicate the status of each type of offline STCU2 internal error condition
- A field that indicates the status of an online hardware abort
- Programmable fault mapping for each BIST for controlling the type of fault that STCU2 reports (recoverable or unrecoverable) when the BIST fails to execute
- Programmable fault mapping for each STCU2 internal error condition for controlling the type of fault that STCU2 reports (recoverable or unrecoverable) when the condition occurs
- Signals for reporting recoverable and unrecoverable faults to the FCCU module
- Redundant recoverable and unrecoverable fault-generation logic to improve reliability
- FCCU recoverable and unrecoverable fault-injection mechanism
- CRCE register access to perform redundant check by software
- LBIST MISRE register access to perform redundant check by software
- Offline BIST-sequence bypass capability
- Global register write-protection mechanism that requires two security key codes
- Global automatic power saving when a BIST sequence is completed, Bypass mode is selected, or a watchdog timer time-out is detected

- ITF/watchdog automatic clock wake-up mechanism when software unlocks the STCU2 registers for write access
- ITF/watchdog automatic power saving when the fixed register-write-access watchdog timer times out

77.4 Block diagram

This diagram shows the parts of STCU2:



This table describes the parts of STCU2:

Part	Function
Reg interface	Provides access to: • Registers • The security key logic • The Internal Peripheral System (IPS) interface • The SSCM Device Configuration Format (DCF) interface
CRC	Performs STCU2 self-checking by sampling a set of selected internal signals when STCU2 is running.
Fault interface	Does the following: • Collects the fault conditions caused by STCU2 internal error conditions and each BIST executed in a sequence.

Table continues on the next page...

Part	Function
	• Depending on the fault mapping for a given BIST, sets the global recoverable or unrecoverable status flag. • Manages the recoverable and unrecoverable fault lines to and from the FCCU module • Manages the set/clear injection mechanism provided by the FCCU module. To improve the intrinsic reliability of this critical logic, the generation logic is duplicated.
Clock interface	Manages the internal and the BIST TCK clock prescaler, the internal clock-gating power saving, and the wake-up clock feature.
WDG FSM (watchdog finite state machine)	Provides the following: • A programmable BIST-sequence-execution watchdog timer that specifies the maximum time allowed for the execution of a BIST sequence • A fixed offline-initialization watchdog timer that specifies the maximum time allowed for offline initialization before indicating an initialization fault • A fixed register-write-access watchdog timer that specifies the maximum amount of time that STCU2 allows you to write to its registers after you unlock them
Master FSM (finite state machine)	Coordinates and schedules all of the operations performed during a BIST sequence.
Loader Shifter FSM (finite state machine)	Programs the BIST registers and reads back the data to be checked at the end of each test operation.
LBIST interface	Provides the interface between the chip's LBIST engines and the STCU2 controllers.
MBIST interface	Provides the interface between the MBIST controller and the STCU2 controllers.

77.5 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed

- The IPS access on a specific register is allowed by software only when the bit STCU2_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following read/write bits: STCU2_CFG.WRP, STCU2_ERR_STAT.UFSF and STCU2_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU2_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on Read/On-Line_Write/Off-Line registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU2_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

NOTE

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

77.6 BISTs and BIST partitions

77.6.1 Definition: BIST

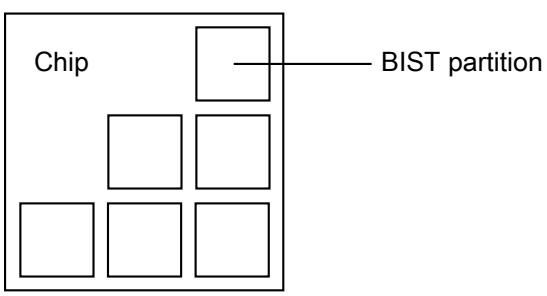
A *built-in self-test (BIST)* is a test that the chip can execute to verify the functional integrity of a part of itself. The chip uses STCU2 and other on-chip hardware to execute a BIST.

77.6.2 Definition: BIST partition

A *built-in self-test (BIST) partition* is a part of a chip for which a BIST has been defined. The hardware that is included in a given BIST partition is chip-specific. For a list of the hardware included in each of the BIST partitions on this chip, see the chip-specific STCU2 information.

77.6.3 Example: BIST partitions on a chip

This is an example of a chip with six built-in self-test (BIST) partitions:



77.6.4 Types of BIST partitions

The STCU2 module on this chip supports the following type or types of built-in self-test (BIST) partitions:

BIST partition type	Description
Memory BIST (MBIST) partition	An SRAM or ROM block
Logic BIST (LBIST) partition	One or more digital modules

77.7 BIST sequences

77.7.1 Definition: BIST sequence

A *built-in self-test (BIST) sequence* is a programmable series of one or more phases, each of which executes one or more individual BISTs.

77.7.2 STCU2 executes MBISTs before LBISTs

If a built-in self-test (BIST) sequence includes both MBISTs and LBISTs, STCU2 executes the MBISTs first.

77.7.3 Example: Single-phase BIST sequence

This is an example of a single-phase BIST sequence in which STCU2 executes only one BIST:

Phase	BIST executed
0	MBIST 16

77.7.4 Example: Multiphase BIST sequence

This is an example of a multiphase BIST sequence in which STCU2 executes more than one BIST in parallel in some phases:

Phase	BISTs executed
0	MBISTs 7, 38
1	MBIST 23
2	MBISTs 6, 16, 29, 23
3	LBIST 12
4	LBISTs 13, 8, 11
5	LBISTs 19, 6

77.7.5 Types of BIST sequences

The STCU2 module on this chip supports these types of built-in self-test (BIST) sequences:

BIST sequence type	Description
Online BIST sequence	A BIST sequence that STCU2 executes during runtime at the request of software. Application software configures and initiates execution of the BIST sequence by loading values into STCU2 registers. Use this type of BIST sequence when you want to test the chip hardware but need to minimize the system startup time. Also use it for failure diagnostics and quality control within a manufacturing environment. You can choose the most appropriate point during runtime at which to execute this type of BIST sequence.
Offline BIST sequence	A BIST sequence that STCU2 automatically executes during reset, either when power is first applied to the chip or when the chip resets STCU2. The chip's SSCM module configures and initiates execution of the BIST sequence by loading values into STCU2 registers from Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory. Use this type of BIST sequence when you want to test the chip hardware before runtime. The location of the DCF record area is chip-specific. The size of the DCF record area is sequence-specific. For the location of the DCF record area, see the chip-specific STCU2 information.

77.7.6 Supported BIST sequences

NXP has validated and therefore supports only specific built-in self-test (BIST) sequences for this chip. For a list of the supported BIST sequences, see the chip-specific STCU2 information.

77.7.7 Definition: Default offline BIST sequence

The *default offline built-in self-test (BIST) sequence* is the offline BIST sequence whose configuration NXP has programmed into the chip's Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory.

If you choose to use the default offline BIST sequence, or if it's the only one supported by NXP, there is no need to create additional DCF records for the offline BIST sequence.

77.8 Functional description

77.8.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/ Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

77.8.2 Reset management

Assertion of the STCU2 reset input signal initializes the STCU2 module status, forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions and on the content of the LBRMSW register.

77.8.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

77.8.2.2 On-Line reset generation

See the chip-specific STCU2 information for details about On-Line reset generation.

77.8.3 Built-in self-test scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST or LBIST depending on the power/timing/ coverage constraints. The limitation in the programming flexibility is described in Design implementation information.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU2_CFG.PTR. The first LBIST is mapped on 00h, the second on 01h, and so on. The first MBIST is mapped to the address 10h, the second to 11h and so on. The additional pointers are in the bitfield STCU2_LB_CTRL[X].PTR for LBIST[X] (where X is the selected LBIST) and STCU2_MB_CTRL[Y].PTR for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU2_LB_CTRL.CSM and STCU2_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the LBIST or MBIST or to close the linked list setting the NIL pointer. For more details, see STCU2 LBIST Control (STCU2_LB_CTRL*n*) and STCU2 MBIST Control Register (STCU2_MB_CTRL*n*) .

NOTE

Program operation should ensure that no flash write/erase operations are ongoing when online LBIST is triggered. Otherwise, a flash segment may be corrupted.

77.8.4 ABORT management

The STCU2 module provides On-Line Self-Test execution abort. When the abort is detected, the On-Line Self-Test operation stops running and the status of the currently running MBIST or LBIST is saved into the related registers to provide intermediate results that might be useful in case of debug or in case of concurrent run. In case of LBIST run, see Reset management, for more details about reset specific management when abort is detected.The following sections describe the differences between these two modalities.

77.8.4.1 Hardware ABORT management

The STCU2 enters the hardware abort condition as a consequence of a Functional reset which causes Self-Test execution to abort .

When hardware abort is detected the bit ABORTHW of the STCU2_ERR_STAT register is set. This flag allows the software to diagnose what has happened during the On-Line Self-Test execution run.

77.8.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU2_RUN[MBPLLEN] and STCU2_RUN[LBPLLEN].

When STCU2_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU2_PLL_CFG register.

When the STCU2_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock signal. As soon as the lock happens, the STCU2 forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one.

Same behavior will be applied when STCU2_RUN[LBPLLEN] is active and LBIST is currently selected.

77.8.6 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

77.8.7 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished in time.

77.8.7.1 Initialization watchdog timer

The STCU2 has two initialization operating modes: program the self-test sequence and run it (Safety mode) or bypass completely the self-test sequence setting the bit STCU2_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded watchdog time-out flags the wrong behavior.

77.8.7.2 Built-in self-test watchdog timer

The LBIST and MBIST execution time has to be configured as described in STCU2 Watchdog Granularity (STCU2_WDG), to account for the overall execution time of the Self-Test sequence. In case the selected LBISTs or MBISTs are not yet completed during the Off-line Self-Test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into STCU2_ERR_STAT.WDTO and STCU2_MBEL/M/H or STCU2_LBE registers while in case of On-Line Self-Test into STCU2_ERR_STAT.WDTOSW and STCU2_MBELSW/MSW/HSW or STCU2_LBESW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in Reset management.

77.8.7.3 Register write-access watchdog timer

As explained in the STCU2_SKC register description:

- A key mechanism protects STCU2 registers during the self-test configuration phase (both online/offline) by preventing any unwanted access.
- A hardware watchdog timer locks register-write access after a number of STCU2 clock cycles. To refresh the hardware watchdog timer before it times out, write only security key 2 to STCU2_SKC.

AUTOLOCK_VALUE is the number of STCU2 clock cycles after which the hardware watchdog timer locks register-write access. See the chip-specific STCU2 information for the value of AUTOLOCK_VALUE.

NOTE

For write completion, each DCF record requires a number of STCU2 clock cycles equal to DCF_COMPLETION. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK_VALUE divided by DCF_COMPLETION. See the chip-specific STCU2 information for the value of DCF_COMPLETION.

The hardware watchdog timer is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration. In this case, the software application might enable write access to the STCU2 registers.

77.8.8 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomial to evaluate the signature of the most important and critical internal signals. This feature is by default switched off but it can be turned on setting the bit STCU2_CFG.CRCEN. Once this bit is set the expected signature has to be programmed into the STCU2_CRCE register; at the end of the Self-Test execution the STCU2 updates the content of the STCU2_CRCR register with the evaluated signature and compares this value with the expected one previously programmed into STCU2_CRCE. In case of mismatches a UF/RF condition is flagged depending on the fault mapping selected into the STCU2_ERR_FM register.

NOTE

The CRC calculation depends on the internal state of the STCU2 when the online/offline self test is started. The STCU2's internal state is dependent on DCF values and their programming sequence for offline selftest. The calculated CRC value can be determined by running an LBIST test under known conditions. The CRC unit produces a stable and predictable result but is dependent on several stimuli (including programming order and the initial internal state of the STCU2 design) during online self test. To calculate a CRC value, run an LBIST test on known good silicon with a predefined software programming sequence. Record the CRC result value. Then provide the STCU2 programming sequence and the initial register vlaues for further runs if the CRC feature is to be used. Once a known good CRC value is determined, the CRC value will not change from one run to another so long as the user does not change the STCU2 programming sequence or any of the

values used to program STCU2 registers. To obtain a consistant CRC calculated value, the STCU must start from the same state compared to the state when the CRC was originally evaluated.

77.8.9 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

77.9 Register description

During Offline SELFTEST, once the STCU2 registers are configured they cannot be overridden.

During Online SELFTEST, once the STCU2 registers are configured they cannot be overridden via IPS till the selftest is complete.

The STCU2 registers are listed in this section.

STCU2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STCU2 Run Register (STCU2_RUN)	32	R/W	0000_0000h	77.9.1/3796
4	STCU2 Run Software Register (STCU2_RUNSW)	32	R/W	0000_0000h	77.9.2/3797
8	STCU2 SK Code Register (STCU2_SKC)	32	W	0000_0000h	77.9.3/3799
C	STCU2 Configuration Register (STCU2_CFG)	32	R/W	See section	77.9.4/3800
10	STCU2 PLL Configuration Register (STCU2_PLL_CFG)	32	R/W	See section	77.9.5/3803
14	STCU2 Watchdog Granularity (STCU2_WDG)	32	R/W	See section	77.9.6/3804
1C	STCU2 CRC Expected Status Register (STCU2_CRCE)	32	R/W	Undefined	77.9.7/3805
20	STCU2 CRC Read Status Register (STCU2_CRCR)	32	R	See section	77.9.8/3806
24	STCU2 Error Register (STCU2_ERR_STAT)	32	R/W	0000_0000h	77.9.9/3807
28	STCU2 Error FM Register (STCU2_ERR_FM)	32	R/W	See section	77.9.10/ 3810
2C	STCU2 Off-Line LBIST Status Register (STCU2_LBS)	32	R	0000_0000h	77.9.11/ 3812
30	STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)	32	R	0000_0000h	77.9.12/ 3813
34	STCU2 On-Line LBIST Status Register (STCU2_LBSSW)	32	R	0000_0000h	77.9.13/ 3814

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38	STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)	32	R	0000_0000h	77.9.14/ 3816
3C	STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)	32	R/W	See section	77.9.15/ 3818
40	STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	32	R/W	See section	77.9.16/ 3820
44	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)	32	R	0000_0000h	77.9.17/ 3821
48	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	32	R	0000_0000h	77.9.18/ 3826
4C	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	32	R	0000_0000h	77.9.19/ 3831
50	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)	32	R	0000_0000h	77.9.20/ 3836
54	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	32	R	0000_0000h	77.9.21/ 3840
58	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	32	R	0000_0000h	77.9.22/ 3843
5C	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)	32	R	0000_0000h	77.9.23/ 3847
60	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	32	R	0000_0000h	77.9.24/ 3850
64	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	32	R	0000_0000h	77.9.25/ 3854
68	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)	32	R	0000_0000h	77.9.26/ 3857
6C	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	32	R	0000_0000h	77.9.27/ 3861
70	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	32	R	0000_0000h	77.9.28/ 3865
74	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	32	R/W	See section	77.9.29/ 3868
78	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	32	R/W	See section	77.9.30/ 3872
7C	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	32	R/W	See section	77.9.31/ 3875
100	STCU2 LBIST Control (STCU2_LB_CTRL0)	32	R/W	See section	77.9.32/ 3878
104	STCU2 LBIST PC Stop Register (STCU2_LB_PCS0)	32	R/W	See section	77.9.33/ 3881
110	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL0)	32	R/W	See section	77.9.34/ 3882
114	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH0)	32	R/W	See section	77.9.35/ 3883

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
118	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL0)	32	R	See section	77.9.36/ 3883
11C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH0)	32	R	See section	77.9.37/ 3884
120	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW0)	32	R/W	See section	77.9.38/ 3885
124	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW0)	32	R/W	See section	77.9.39/ 3886
128	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW0)	32	R	See section	77.9.40/ 3886
12C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW0)	32	R	See section	77.9.41/ 3887
140	STCU2 LBIST Control (STCU2_LB_CTRL1)	32	R/W	See section	77.9.32/ 3878
144	STCU2 LBIST PC Stop Register (STCU2_LB_PCS1)	32	R/W	See section	77.9.33/ 3881
150	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL1)	32	R/W	See section	77.9.34/ 3882
154	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH1)	32	R/W	See section	77.9.35/ 3883
158	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL1)	32	R	See section	77.9.36/ 3883
15C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH1)	32	R	See section	77.9.37/ 3884
160	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW1)	32	R/W	See section	77.9.38/ 3885
164	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW1)	32	R/W	See section	77.9.39/ 3886
168	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW1)	32	R	See section	77.9.40/ 3886
16C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW1)	32	R	See section	77.9.41/ 3887
180	STCU2 LBIST Control (STCU2_LB_CTRL2)	32	R/W	See section	77.9.32/ 3878
184	STCU2 LBIST PC Stop Register (STCU2_LB_PCS2)	32	R/W	See section	77.9.33/ 3881
190	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL2)	32	R/W	See section	77.9.34/ 3882
194	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH2)	32	R/W	See section	77.9.35/ 3883
198	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL2)	32	R	See section	77.9.36/ 3883
19C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH2)	32	R	See section	77.9.37/ 3884

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW2)	32	R/W	See section	77.9.38/ 3885
1A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW2)	32	R/W	See section	77.9.39/ 3886
1A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW2)	32	R	See section	77.9.40/ 3886
1AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW2)	32	R	See section	77.9.41/ 3887
1C0	STCU2 LBIST Control (STCU2_LB_CTRL3)	32	R/W	See section	77.9.32/ 3878
1C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS3)	32	R/W	See section	77.9.33/ 3881
1D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL3)	32	R/W	See section	77.9.34/ 3882
1D4	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH3)	32	R/W	See section	77.9.35/ 3883
1D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL3)	32	R	See section	77.9.36/ 3883
1DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH3)	32	R	See section	77.9.37/ 3884
1E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW3)	32	R/W	See section	77.9.38/ 3885
1E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW3)	32	R/W	See section	77.9.39/ 3886
1E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW3)	32	R	See section	77.9.40/ 3886
1EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW3)	32	R	See section	77.9.41/ 3887
200	STCU2 LBIST Control (STCU2_LB_CTRL4)	32	R/W	See section	77.9.32/ 3878
204	STCU2 LBIST PC Stop Register (STCU2_LB_PCS4)	32	R/W	See section	77.9.33/ 3881
210	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL4)	32	R/W	See section	77.9.34/ 3882
214	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH4)	32	R/W	See section	77.9.35/ 3883
218	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL4)	32	R	See section	77.9.36/ 3883
21C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH4)	32	R	See section	77.9.37/ 3884
220	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW4)	32	R/W	See section	77.9.38/ 3885
224	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW4)	32	R/W	See section	77.9.39/ 3886

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
228	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW4)	32	R	See section	77.9.40/ 3886
22C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW4)	32	R	See section	77.9.41/ 3887
240	STCU2 LBIST Control (STCU2_LB_CTRL5)	32	R/W	See section	77.9.32/ 3878
244	STCU2 LBIST PC Stop Register (STCU2_LB_PCS5)	32	R/W	See section	77.9.33/ 3881
250	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL5)	32	R/W	See section	77.9.34/ 3882
254	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH5)	32	R/W	See section	77.9.35/ 3883
258	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL5)	32	R	See section	77.9.36/ 3883
25C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH5)	32	R	See section	77.9.37/ 3884
260	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW5)	32	R/W	See section	77.9.38/ 3885
264	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW5)	32	R/W	See section	77.9.39/ 3886
268	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW5)	32	R	See section	77.9.40/ 3886
26C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW5)	32	R	See section	77.9.41/ 3887
280	STCU2 LBIST Control (STCU2_LB_CTRL6)	32	R/W	See section	77.9.32/ 3878
284	STCU2 LBIST PC Stop Register (STCU2_LB_PCS6)	32	R/W	See section	77.9.33/ 3881
290	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL6)	32	R/W	See section	77.9.34/ 3882
294	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH6)	32	R/W	See section	77.9.35/ 3883
298	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL6)	32	R	See section	77.9.36/ 3883
29C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH6)	32	R	See section	77.9.37/ 3884
2A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW6)	32	R/W	See section	77.9.38/ 3885
2A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW6)	32	R/W	See section	77.9.39/ 3886
2A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW6)	32	R	See section	77.9.40/ 3886
2AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW6)	32	R	See section	77.9.41/ 3887

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C0	STCU2 LBIST Control (STCU2_LB_CTRL7)	32	R/W	See section	77.9.32/ 3878
2C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS7)	32	R/W	See section	77.9.33/ 3881
2D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL7)	32	R/W	See section	77.9.34/ 3882
2D4	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH7)	32	R/W	See section	77.9.35/ 3883
2D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL7)	32	R	See section	77.9.36/ 3883
2DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH7)	32	R	See section	77.9.37/ 3884
2E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW7)	32	R/W	See section	77.9.38/ 3885
2E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW7)	32	R/W	See section	77.9.39/ 3886
2E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW7)	32	R	See section	77.9.40/ 3886
2EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW7)	32	R	See section	77.9.41/ 3887
600	STCU2 MBIST Control Register (STCU2_MB_CTRL0)	32	R/W	See section	77.9.42/ 3888
604	STCU2 MBIST Control Register (STCU2_MB_CTRL1)	32	R/W	See section	77.9.42/ 3888
608	STCU2 MBIST Control Register (STCU2_MB_CTRL2)	32	R/W	See section	77.9.42/ 3888
60C	STCU2 MBIST Control Register (STCU2_MB_CTRL3)	32	R/W	See section	77.9.42/ 3888
610	STCU2 MBIST Control Register (STCU2_MB_CTRL4)	32	R/W	See section	77.9.42/ 3888
614	STCU2 MBIST Control Register (STCU2_MB_CTRL5)	32	R/W	See section	77.9.42/ 3888
618	STCU2 MBIST Control Register (STCU2_MB_CTRL6)	32	R/W	See section	77.9.42/ 3888
61C	STCU2 MBIST Control Register (STCU2_MB_CTRL7)	32	R/W	See section	77.9.42/ 3888
620	STCU2 MBIST Control Register (STCU2_MB_CTRL8)	32	R/W	See section	77.9.42/ 3888
624	STCU2 MBIST Control Register (STCU2_MB_CTRL9)	32	R/W	See section	77.9.42/ 3888
628	STCU2 MBIST Control Register (STCU2_MB_CTRL10)	32	R/W	See section	77.9.42/ 3888
62C	STCU2 MBIST Control Register (STCU2_MB_CTRL11)	32	R/W	See section	77.9.42/ 3888

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
630	STCU2 MBIST Control Register (STCU2_MB_CTRL12)	32	R/W	See section	77.9.42/ 3888
634	STCU2 MBIST Control Register (STCU2_MB_CTRL13)	32	R/W	See section	77.9.42/ 3888
638	STCU2 MBIST Control Register (STCU2_MB_CTRL14)	32	R/W	See section	77.9.42/ 3888
63C	STCU2 MBIST Control Register (STCU2_MB_CTRL15)	32	R/W	See section	77.9.42/ 3888
640	STCU2 MBIST Control Register (STCU2_MB_CTRL16)	32	R/W	See section	77.9.42/ 3888
644	STCU2 MBIST Control Register (STCU2_MB_CTRL17)	32	R/W	See section	77.9.42/ 3888
648	STCU2 MBIST Control Register (STCU2_MB_CTRL18)	32	R/W	See section	77.9.42/ 3888
64C	STCU2 MBIST Control Register (STCU2_MB_CTRL19)	32	R/W	See section	77.9.42/ 3888
650	STCU2 MBIST Control Register (STCU2_MB_CTRL20)	32	R/W	See section	77.9.42/ 3888
654	STCU2 MBIST Control Register (STCU2_MB_CTRL21)	32	R/W	See section	77.9.42/ 3888
658	STCU2 MBIST Control Register (STCU2_MB_CTRL22)	32	R/W	See section	77.9.42/ 3888
65C	STCU2 MBIST Control Register (STCU2_MB_CTRL23)	32	R/W	See section	77.9.42/ 3888
660	STCU2 MBIST Control Register (STCU2_MB_CTRL24)	32	R/W	See section	77.9.42/ 3888
664	STCU2 MBIST Control Register (STCU2_MB_CTRL25)	32	R/W	See section	77.9.42/ 3888
668	STCU2 MBIST Control Register (STCU2_MB_CTRL26)	32	R/W	See section	77.9.42/ 3888
66C	STCU2 MBIST Control Register (STCU2_MB_CTRL27)	32	R/W	See section	77.9.42/ 3888
670	STCU2 MBIST Control Register (STCU2_MB_CTRL28)	32	R/W	See section	77.9.42/ 3888
674	STCU2 MBIST Control Register (STCU2_MB_CTRL29)	32	R/W	See section	77.9.42/ 3888
678	STCU2 MBIST Control Register (STCU2_MB_CTRL30)	32	R/W	See section	77.9.42/ 3888
67C	STCU2 MBIST Control Register (STCU2_MB_CTRL31)	32	R/W	See section	77.9.42/ 3888
680	STCU2 MBIST Control Register (STCU2_MB_CTRL32)	32	R/W	See section	77.9.42/ 3888
684	STCU2 MBIST Control Register (STCU2_MB_CTRL33)	32	R/W	See section	77.9.42/ 3888

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
688	STCU2 MBIST Control Register (STCU2_MB_CTRL34)	32	R/W	See section	77.9.42/ 3888
68C	STCU2 MBIST Control Register (STCU2_MB_CTRL35)	32	R/W	See section	77.9.42/ 3888
690	STCU2 MBIST Control Register (STCU2_MB_CTRL36)	32	R/W	See section	77.9.42/ 3888
694	STCU2 MBIST Control Register (STCU2_MB_CTRL37)	32	R/W	See section	77.9.42/ 3888
698	STCU2 MBIST Control Register (STCU2_MB_CTRL38)	32	R/W	See section	77.9.42/ 3888
69C	STCU2 MBIST Control Register (STCU2_MB_CTRL39)	32	R/W	See section	77.9.42/ 3888
6A0	STCU2 MBIST Control Register (STCU2_MB_CTRL40)	32	R/W	See section	77.9.42/ 3888
6A4	STCU2 MBIST Control Register (STCU2_MB_CTRL41)	32	R/W	See section	77.9.42/ 3888
6A8	STCU2 MBIST Control Register (STCU2_MB_CTRL42)	32	R/W	See section	77.9.42/ 3888
6AC	STCU2 MBIST Control Register (STCU2_MB_CTRL43)	32	R/W	See section	77.9.42/ 3888
6B0	STCU2 MBIST Control Register (STCU2_MB_CTRL44)	32	R/W	See section	77.9.42/ 3888
6B4	STCU2 MBIST Control Register (STCU2_MB_CTRL45)	32	R/W	See section	77.9.42/ 3888
6B8	STCU2 MBIST Control Register (STCU2_MB_CTRL46)	32	R/W	See section	77.9.42/ 3888
6BC	STCU2 MBIST Control Register (STCU2_MB_CTRL47)	32	R/W	See section	77.9.42/ 3888
6C0	STCU2 MBIST Control Register (STCU2_MB_CTRL48)	32	R/W	See section	77.9.42/ 3888
6C4	STCU2 MBIST Control Register (STCU2_MB_CTRL49)	32	R/W	See section	77.9.42/ 3888
6C8	STCU2 MBIST Control Register (STCU2_MB_CTRL50)	32	R/W	See section	77.9.42/ 3888
6CC	STCU2 MBIST Control Register (STCU2_MB_CTRL51)	32	R/W	See section	77.9.42/ 3888
6D0	STCU2 MBIST Control Register (STCU2_MB_CTRL52)	32	R/W	See section	77.9.42/ 3888
6D4	STCU2 MBIST Control Register (STCU2_MB_CTRL53)	32	R/W	See section	77.9.42/ 3888
6D8	STCU2 MBIST Control Register (STCU2_MB_CTRL54)	32	R/W	See section	77.9.42/ 3888
6DC	STCU2 MBIST Control Register (STCU2_MB_CTRL55)	32	R/W	See section	77.9.42/ 3888

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6E0	STCU2 MBIST Control Register (STCU2_MB_CTRL56)	32	R/W	See section	77.9.42/ 3888
6E4	STCU2 MBIST Control Register (STCU2_MB_CTRL57)	32	R/W	See section	77.9.42/ 3888
6E8	STCU2 MBIST Control Register (STCU2_MB_CTRL58)	32	R/W	See section	77.9.42/ 3888
6EC	STCU2 MBIST Control Register (STCU2_MB_CTRL59)	32	R/W	See section	77.9.42/ 3888
6F0	STCU2 MBIST Control Register (STCU2_MB_CTRL60)	32	R/W	See section	77.9.42/ 3888
6F4	STCU2 MBIST Control Register (STCU2_MB_CTRL61)	32	R/W	See section	77.9.42/ 3888
6F8	STCU2 MBIST Control Register (STCU2_MB_CTRL62)	32	R/W	See section	77.9.42/ 3888
6FC	STCU2 MBIST Control Register (STCU2_MB_CTRL63)	32	R/W	See section	77.9.42/ 3888
700	STCU2 MBIST Control Register (STCU2_MB_CTRL64)	32	R/W	See section	77.9.42/ 3888
704	STCU2 MBIST Control Register (STCU2_MB_CTRL65)	32	R/W	See section	77.9.42/ 3888
708	STCU2 MBIST Control Register (STCU2_MB_CTRL66)	32	R/W	See section	77.9.42/ 3888
70C	STCU2 MBIST Control Register (STCU2_MB_CTRL67)	32	R/W	See section	77.9.42/ 3888
710	STCU2 MBIST Control Register (STCU2_MB_CTRL68)	32	R/W	See section	77.9.42/ 3888
714	STCU2 MBIST Control Register (STCU2_MB_CTRL69)	32	R/W	See section	77.9.42/ 3888
718	STCU2 MBIST Control Register (STCU2_MB_CTRL70)	32	R/W	See section	77.9.42/ 3888
71C	STCU2 MBIST Control Register (STCU2_MB_CTRL71)	32	R/W	See section	77.9.42/ 3888
720	STCU2 MBIST Control Register (STCU2_MB_CTRL72)	32	R/W	See section	77.9.42/ 3888
724	STCU2 MBIST Control Register (STCU2_MB_CTRL73)	32	R/W	See section	77.9.42/ 3888
728	STCU2 MBIST Control Register (STCU2_MB_CTRL74)	32	R/W	See section	77.9.42/ 3888
72C	STCU2 MBIST Control Register (STCU2_MB_CTRL75)	32	R/W	See section	77.9.42/ 3888
730	STCU2 MBIST Control Register (STCU2_MB_CTRL76)	32	R/W	See section	77.9.42/ 3888
734	STCU2 MBIST Control Register (STCU2_MB_CTRL77)	32	R/W	See section	77.9.42/ 3888

Table continues on the next page...

STCU2 memory map (continued)

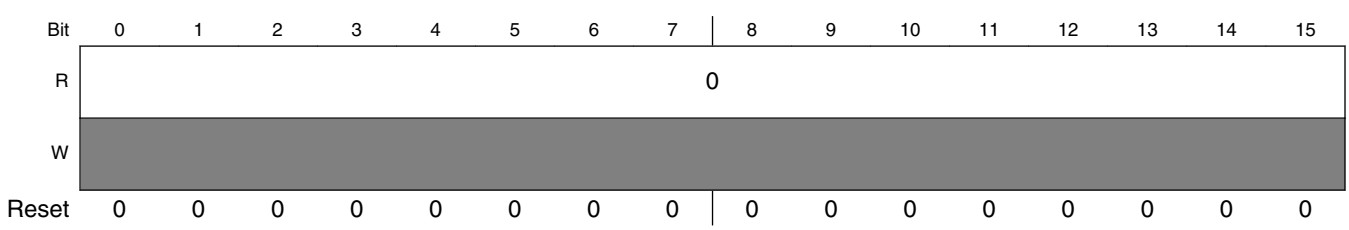
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
738	STCU2 MBIST Control Register (STCU2_MB_CTRL78)	32	R/W	See section	77.9.42/ 3888
73C	STCU2 MBIST Control Register (STCU2_MB_CTRL79)	32	R/W	See section	77.9.42/ 3888
740	STCU2 MBIST Control Register (STCU2_MB_CTRL80)	32	R/W	See section	77.9.42/ 3888
744	STCU2 MBIST Control Register (STCU2_MB_CTRL81)	32	R/W	See section	77.9.42/ 3888
748	STCU2 MBIST Control Register (STCU2_MB_CTRL82)	32	R/W	See section	77.9.42/ 3888
74C	STCU2 MBIST Control Register (STCU2_MB_CTRL83)	32	R/W	See section	77.9.42/ 3888
750	STCU2 MBIST Control Register (STCU2_MB_CTRL84)	32	R/W	See section	77.9.42/ 3888
754	STCU2 MBIST Control Register (STCU2_MB_CTRL85)	32	R/W	See section	77.9.42/ 3888
758	STCU2 MBIST Control Register (STCU2_MB_CTRL86)	32	R/W	See section	77.9.42/ 3888
75C	STCU2 MBIST Control Register (STCU2_MB_CTRL87)	32	R/W	See section	77.9.42/ 3888
760	STCU2 MBIST Control Register (STCU2_MB_CTRL88)	32	R/W	See section	77.9.42/ 3888
764	STCU2 MBIST Control Register (STCU2_MB_CTRL89)	32	R/W	See section	77.9.42/ 3888
768	STCU2 MBIST Control Register (STCU2_MB_CTRL90)	32	R/W	See section	77.9.42/ 3888
76C	STCU2 MBIST Control Register (STCU2_MB_CTRL91)	32	R/W	See section	77.9.42/ 3888

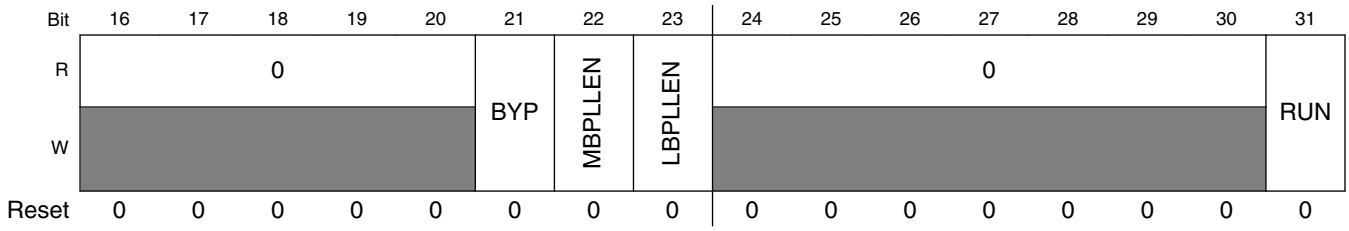
77.9.1 STCU2 Run Register (STCU2_RUN)

The STCU2_RUN register defines the RUN bit to start the off-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 0h offset = 0h





STCU2_RUN field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BYP	Bypass mode 0 Off-Line Self-Test is active. In case the RUN bit into the STCU2_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU2_ERR_STAT register. 1 Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU2_SKC register is written according to the request access mode.
22 MBPLLEN	Off-Line MBIST with PLL Enabled 0 Off-Line MBIST is executed without using the on-chip PLL. 1 Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
23 LBPLLEN	Off-Line LBIST with PLL Enabled 0 Off-Line LBIST is executed without using the on-chip PLL. 1 Off-Line LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUN	RUN: RUN The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed. 0 Idle 1 Off-Line Self testing procedure is running

77.9.2 STCU2 Run Software Register (STCU2_RUNSW)

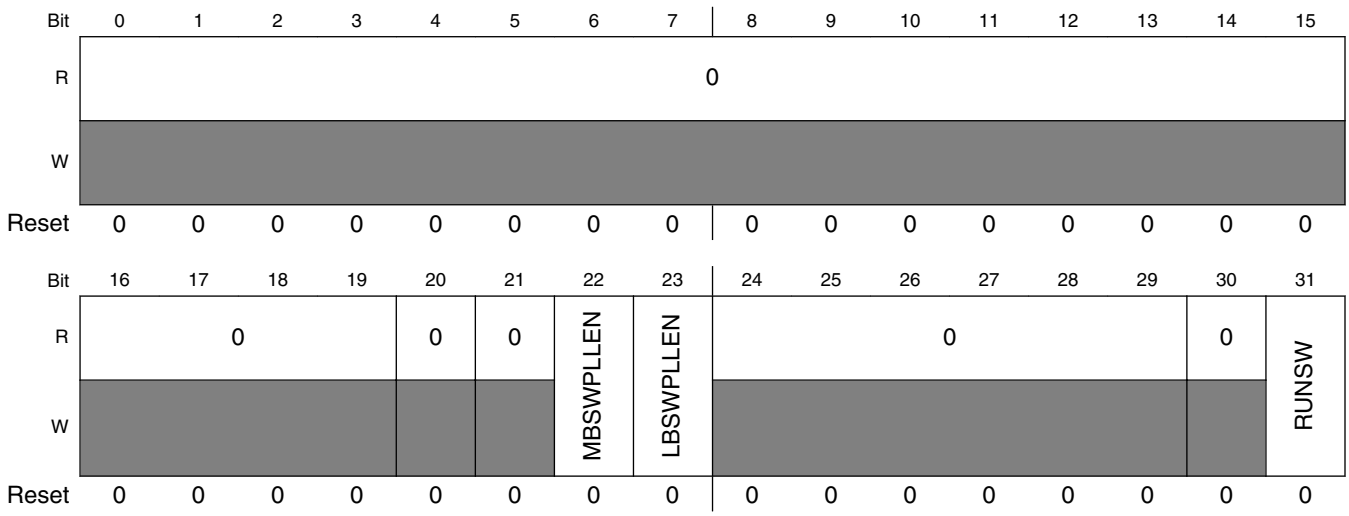
The STCU2_RUNSW register defines the RUN bit to start the on-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU2_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU2

Register description

Address: 0h base + 4h offset = 4h



STCU2_RUNSW field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 MBSWPLLEN	On-Line MBIST with PLL Enabled 0 On-Line MBIST is executed without using the on-chip PLL. 1 On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
23 LBSWPLLEN	On-Line LBIST with PLL Enabled 0 On-Line LBIST is executed without using the on-chip PLL. 1 On-Line LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed. 0 Idle 1 On-line self testing procedure is running

77.9.3 STCU2 SK Code Register (STCU2_SKC)

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus)or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM busis asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access lasts more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU2_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SKC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_SKC field descriptions

Field	Description
0–31 SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2 STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

77.9.4 STCU2 Configuration Register (STCU2_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

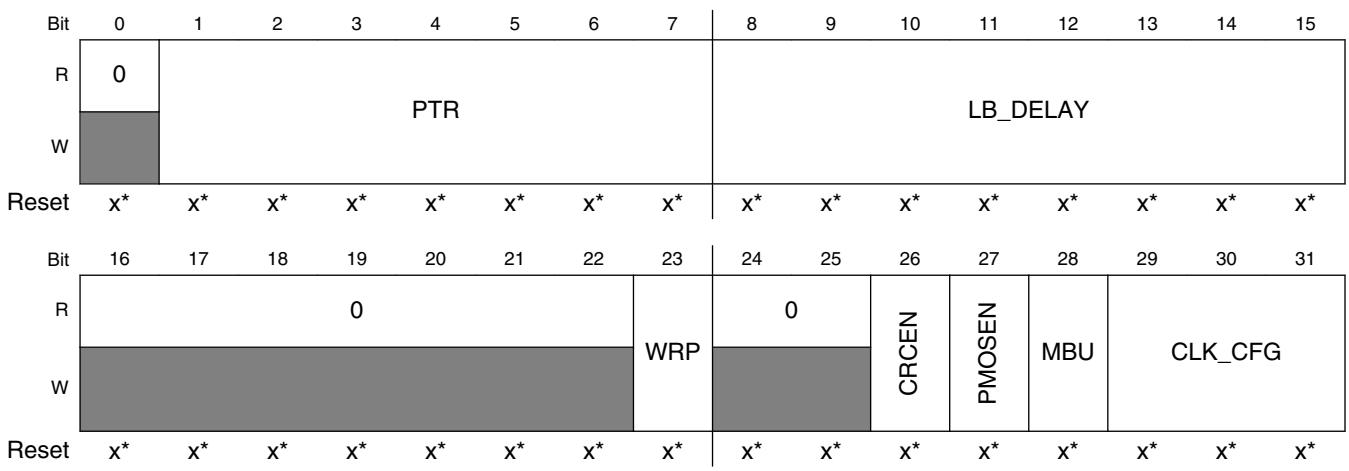
MBIST can be run in three different test types according to how the STCU2_CFG register bits 27 and 28 are set.

The following table describes the test algorithm and coverage of the different test types.

Table 77-4. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_CFG[PMOSEN]	STCU2_CFG[MBU]	MBIST Algorithm	Internal Nodes/ Circuits Additionally Covered by Test	Use Case	Test Cycle
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by MCU production test. Recommended to use in the field for MBIST online self-test.	About 3 to 4x more than Auto Test
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm.	Address Decoder and Bitcell	Recommended to use in the field for MBIST online self-test if the Full Test takes too long.	About 10% less than Full Test
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self-test as an optimum balance of test time vs. fault coverage.	About 3 to 4x faster than Full Test

Address: 0h base + Ch offset = Ch



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_CFG field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	LBIST or MBIST pointer PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
	You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
8–15 LB_DELAY	Delay LBIST run LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. The allowed delay time are the following: 00h: No delay 01h: 1 x16 STCU2 Core clock cycles 02h: 2 x 16 STCU2 Core clock cycles 03h: 3 x 16 STCU2 Core clock cycles ... FDh: 253 x 16 STCU2 Core clock cycles FEh: 254 x 16 STCU2 Core clock cycles FFh: 255 x 16 STCU2 Core clock cycles Note that in case the Frequency of the STCU2 core clock is 16 MHz, the delay parameters is expressed in µs. **NOTE:** This field should be written with 0xFF for self test run. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 CRCEN	CRCEN: CRC Enable comparison You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0 The CRC comparison is not performed and the STCU2_ERR_STAT.CRCS[SW] and STCU2_ERR_STAT.UF/RF global flags are not updated in case of mismatches between STCU2.CRCE and STCU2.CRCR values 1 The CRC comparison is performed and the status is updated into the related flags of the STCU2_ERR_STAT register
27 PMOSEN	MBIST PMOS Test Enable When MBU is 1 then "Auto test" MBIST test type will be selected regardless of the setting of this field.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
	0 When MBU is 0 then selects the "Reduced test" MBIST test type. 1 When MBU is 0 then selects the "Full test" MBIST test type.
28 MBU	MBIST MBU Test Enabled 0 Either "Full test" or "Reduced test" MBIST test type will be selected according to the PMOSEN value. 1 Selects "Auto test" MBIST test type regardless of the PMOSEN value.
29–31 CLK_CFG	Logic, Memory BIST and STCU2 core CLK Clock configuration CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following: 000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8 You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

77.9.5 STCU2 PLL Configuration Register (STCU2_PLL_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU2_RUN[MBPLLEN] = 1 or STCU2_RUN[LBPLLEN] = 1.

In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		PLLODF						0					PLLIDF			0									PLLLDF						
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_PLL_CFG field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 PLLODF	PLL Output Division Factor The value of this field drives the Output Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the related PLL's Output Division factor.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 PLLIDF	PLL Input Division Factor The value of this field drives the Input Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the related PLL's Input Division factor.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PLLLDF	PLL Loop Division Factor The value of this field drives the Loop Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the related PLL's Loop Division factor.

77.9.6 STCU2 Watchdog Granularity (STCU2_WDG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_WDG register defines the time limit for LBIST and MBIST execution, providing a protection mechanism in case of deadlock or endless conditions during the self test procedure.

When the offline self test sequence is not run and the STCU2_CFG[BYP] field is not set, bits 15..0 define the time-out before the STCU2_ERR[WDTO] field is set and the STCU2 core clock is switched off.

In case offline/online self test sequence is run, the STCU2_WDG value defines the time-out of the execution run.

The register fields are always readable. You can write to these fields when either of the following conditions is true:

- Offline self test phase is active
- Online self test phase is active and CFG[WRP] = 0

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

STCU2_WDG field descriptions

Field	Description
0–31 WDGEOC	Watchdog End of Count Timer This value defines the time limit for offline and online self test execution to ensure correct operation within the allowed time. **NOTE:** If this register is programmed while STCU2_RUN[RUN]=1, the new STCU2_WDG value, programmed after STCU2_RUN, does not take effect. The allowed values are: 0000 0000h: 1 × 16 STCU2 Core clock cycles 0000 0001h: 2 × 16 STCU2 Core clock cycles 0000 0002h: 3 × 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 × 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 × 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 × 16 STCU2 Core clock cycles

77.9.7 STCU2 CRC Expected Status Register (STCU2_CRCE)

NOTE

See the chip specific information for the default reset value of this field.

The STCU2_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU2_CRCE_RES and computes the CRC signature of the STCU2's critical signals and when STCU2_CFG[CRCEN] = 1, it is able to provide the Self-Check capability of the STCU2 managing the CRCS[SW] bits of the STCU2_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

CRC-32 [ethernet protocol]

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRCE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

STCU2_CRCE field descriptions

Field	Description
0–31 CRCE	CRC expected signature

77.9.8 STCU2 CRC Read Status Register (STCU2_CRCR)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CRCR register reports the value obtained at the end of the Off/On-line Self-Test sequence. It might be used for diagnoses and also as additional check with respect to the CRCS[SW] bit of the STCU2_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCR																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

STCU2_CRCR field descriptions

Field	Description
0–31 CRCR	Read CRC signature

77.9.9 STCU2 Error Register (STCU2_ERR_STAT)

The STCU2_ERR_STAT register includes the status flags related to the STCU2 internal fault conditions occurred during the configuration or the On/Off-Line self testing execution.

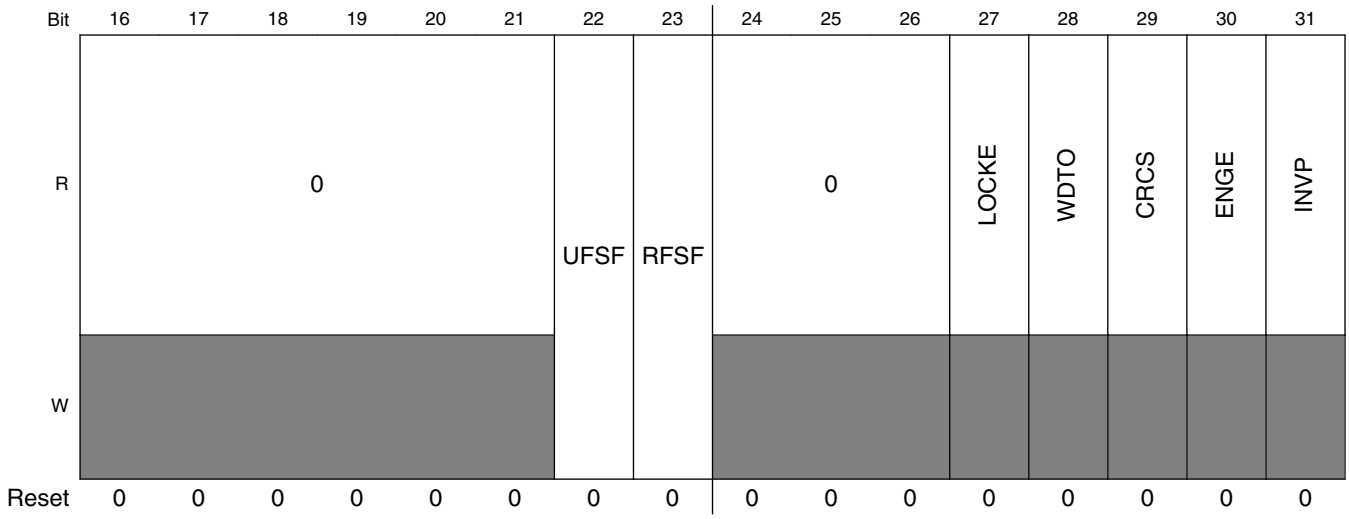
The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						ABORTHW	0	0			LOCKESW	WDTOSW	CRCSSW	ENGESW	INVPSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



STCU2_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ABORTHW	On-line hardware abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No hardware abort was requested during the On-Life Self-Test sequence 1 A hardware abort was detected during the On-Life Self-Test sequence
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	On-Line LOCK Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 In case PLL is enabled, it is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUNSW.LBSWPLLEN is set and/or during the MBIST run when STCU2_RUNSW[MBSWPLLEN] = 1.
12 WDTOSW	On-Line Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
13 CRCSSW	On-Line CRC Status This flag is activated ONLY when On-Line Self Test is performed and STCU2_CFG.CRCEN is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU2_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU2_CRCR register report the current evaluated CRC. You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Successful CRC comparison or comparison has been masked (STCU2_CFG.CRCEN set to 0) 1 Failed CRC comparison
14 ENGESW	On-Line Engine Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	On-Line Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Unrecoverable Faults condition 1 There are errors that trigger the Unrecoverable Faults condition
23 RFSF	Recoverable Faults Status Flag This flag reports the global status of the RF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Recoverable Faults condition 1 There are errors that trigger the Recoverable Faults condition
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Off-Line LOCK Error 0 The PLL is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The Off-Line Self-test run is stopped and the status

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUN[LBPLLEN] = 1 and/or during the MBIST run when STCU2_RUN[MBPLLEN] = 1.
28 WDTO	Off-Line Watchdog time-out 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The STCU2_RUN[RUN] bit or the STCU2_RUN[BYP] bit is not set before the Watchdog reaches the End of Count • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 CRCS	Off-Line CRC status This flag is activated ONLY when Off-Line Self Test is performed STCU2_CFG[CRCEN] is set to 1 otherwise it is always forced to 0 preventing in such a way the generation of the related STCU2_ERR_STAT.UFSF/RFSF bits. In both the cases, the content of the STCU2_CRCR register report the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU2_CFG[CRCEN] programmed to 0) 1 Failed CRC comparison
30 ENGE	Off-Line Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Off-Line Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)

77.9.10 STCU2 Error FM Register (STCU2_ERR_FM)

NOTE

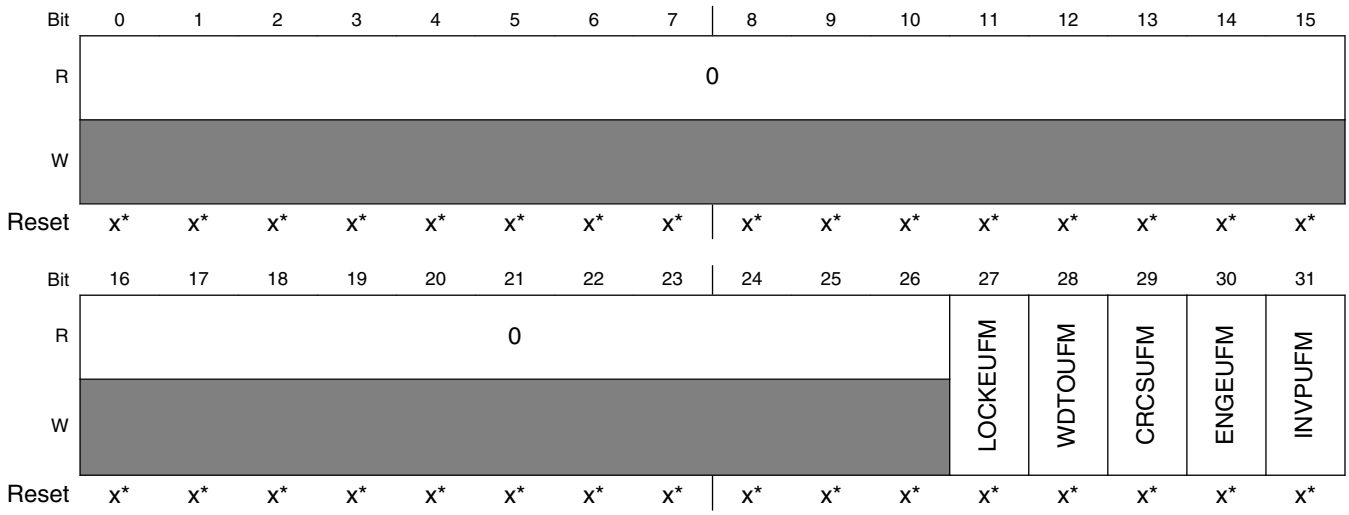
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU2_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 28h offset = 28h



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_ERR_FM field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
28 WDTOUFM	Watchdog time-out Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
29 CRCSUFM	CRC Status Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_ERR_FM field descriptions (continued)

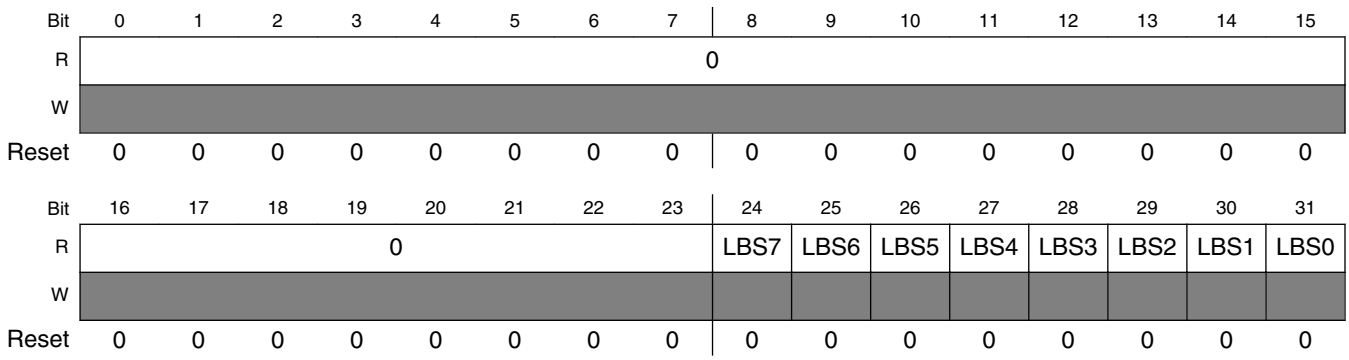
Field	Description
	0 Recoverable Fault Mapping 1 Unrecoverable Mapping

77.9.11 STCU2 Off-Line LBIST Status Register (STCU2_LBS)

The STCU2_LBS register includes the results corresponding to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 2Ch offset = 2Ch



STCU2_LBS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LBS7	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
25 LBS6	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
26 LBS5	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBS4	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBS3	Off-Line status of the selected LBIST

Table continues on the next page...

STCU2_LBS field descriptions (continued)

Field	Description
	0 Failed LBIST execution 1 Successful LBIST execution
29 LBS2	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBS1	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBS0	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

77.9.12 STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)

The STCU2_LBE register includes the End Flag related to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LBE7	LBE6	LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBE field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LBE7	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
25 LBE6	Off-Line End status of the selected LBIST

Table continues on the next page...

STCU2_LBE field descriptions (continued)

Field	Description
	0 LBIST execution not yet completed 1 LBIST execution finished
26 LBE5	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBE4	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBE3	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBE2	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBE1	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBE0	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

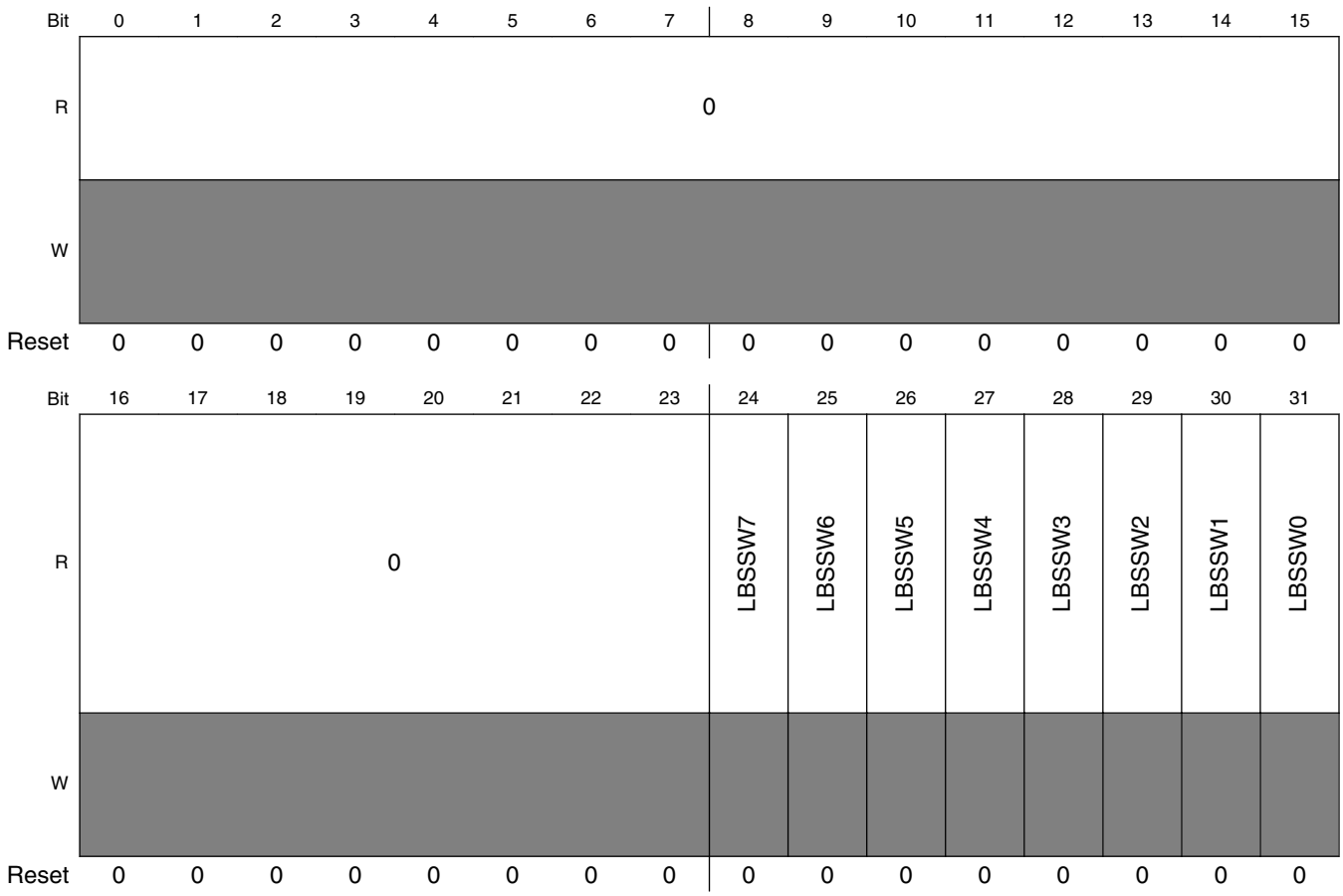
77.9.13 STCU2 On-Line LBIST Status Register (STCU2_LBSSW)

The STCU2_LBSSW register includes the results corresponding to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 34h offset = 34h



STCU2_LBSSW field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LBSSW7	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
25 LBSSW6	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
26 LBSSW5	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBSSW4	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBSSW3	LBSSWx: On-Line status of the selected LBIST

Table continues on the next page...

STCU2_LBSSW field descriptions (continued)

Field	Description
	0 Failed LBIST execution 1 Successful LBIST execution
29 LBSSW2	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBSSW1	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBSSW0	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

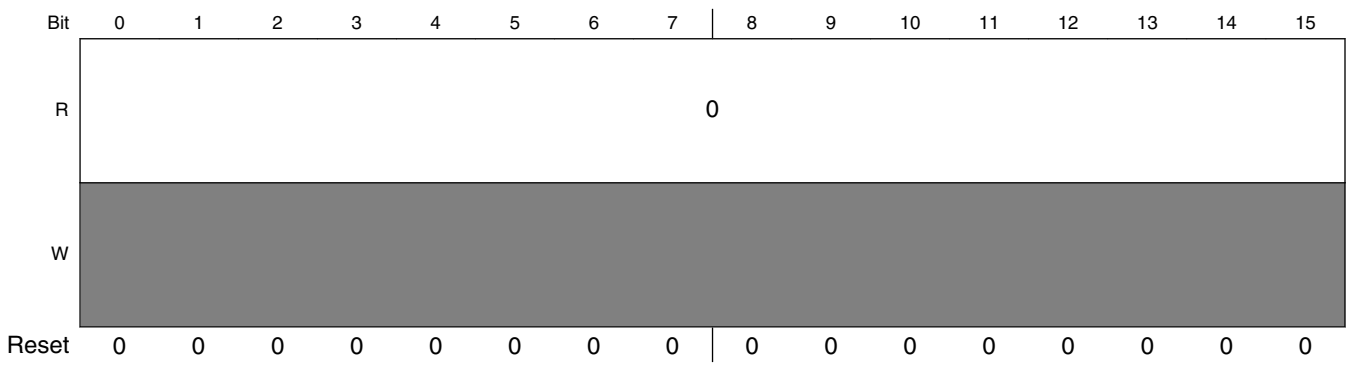
77.9.14 STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)

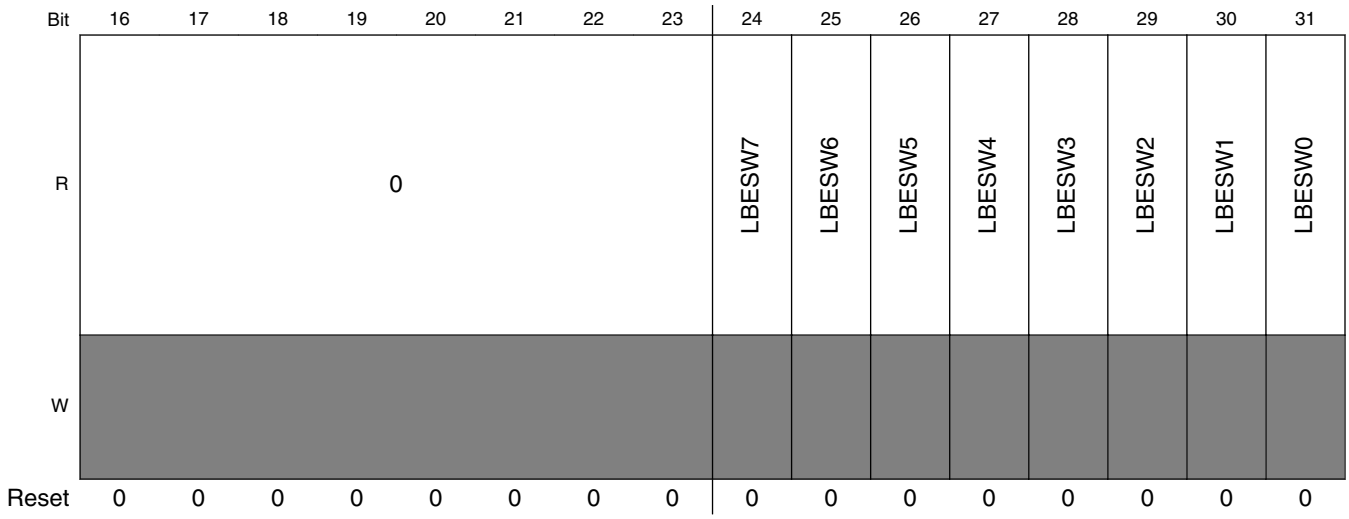
The STCU2_LBESW register includes the End Flag related to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 38h offset = 38h





STCU2_LBESW field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LBESW7	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
25 LBESW6	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
26 LBESW5	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBESW4	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBESW3	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBESW2	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBESW1	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBESW0	LBESWx: On-Line LBIST End status

Table continues on the next page...

STCU2_LBESW field descriptions (continued)

Field	Description
	0 LBIST execution not yet completed 1 LBIST execution finished

77.9.15 STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBRMSW register defines the On-Line reset mapping for each LBIST. Global reset can be programmed.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LBRMSW7	LBRMSW6	LBRMSW5	LBRMSW4	LBRMSW3	LBRMSW2	LBRMSW1	LBRMSW0
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBRMSW field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LBRMSW7	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
25 LBRMSW6	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
26 LBRMSW5	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
27 LBRMSW4	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
28 LBRMSW3	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
29 LBRMSW2	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
30 LBRMSW1	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run
31 LBRMSW0	On-Line LBIST Reset Management

Table continues on the next page...

STCU2_LBRMSW field descriptions (continued)

Field	Description
	NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Reserved 1 Global functional reset is pulsed at the end of LBIST run

77.9.16 STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBUFM register defines the fault mapping of each LBIST in terms of Unrecoverable or Recoverable Fault.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W	Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W	Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBUFM field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 LBUFM7	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 LBUFM6	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 LBUFM5	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 LBUFM4	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 LBUFM3	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 LBUFM2	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 LBUFM1	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 LBUFM0	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

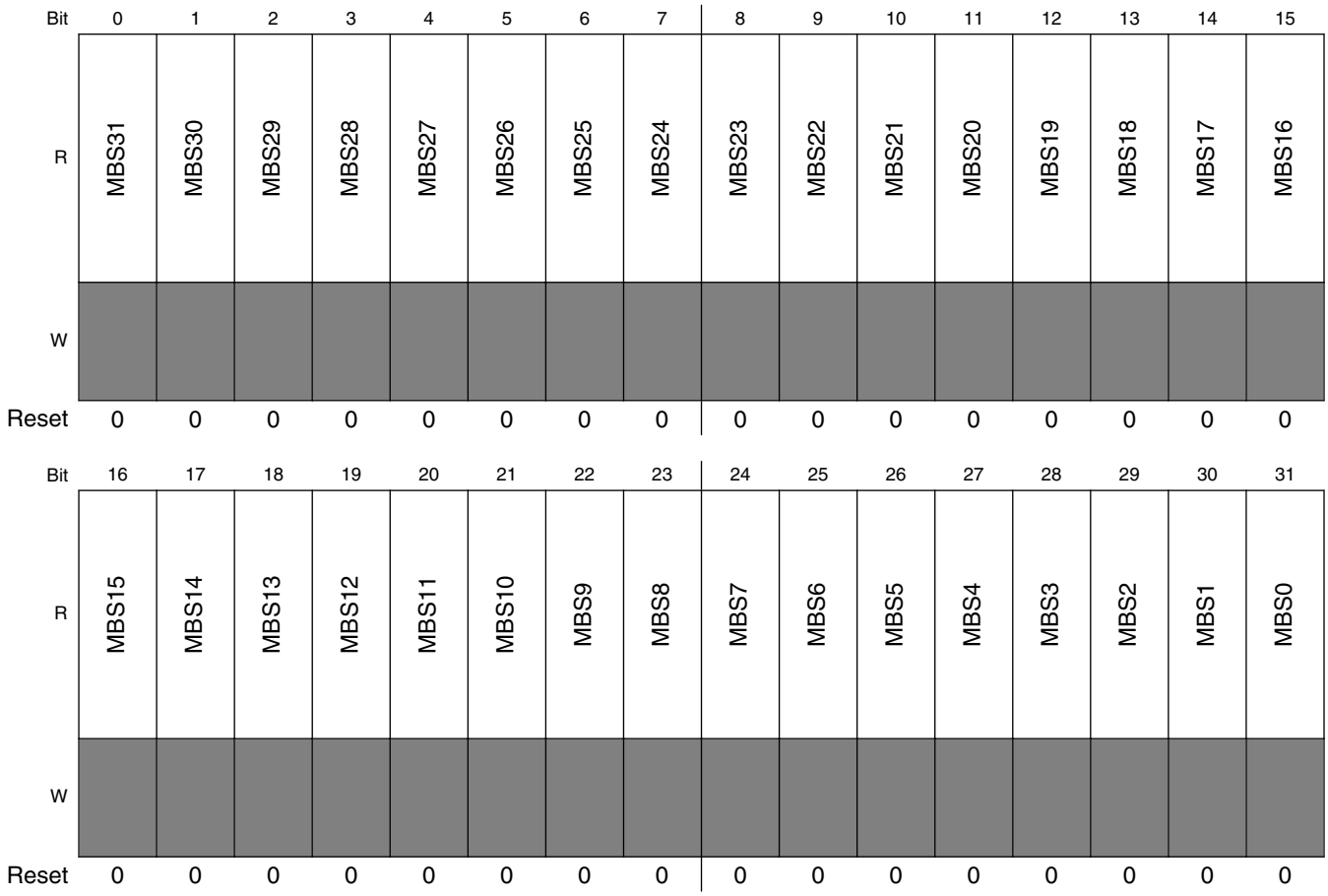
77.9.17 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)

The STCU2_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs .

Register description

Address: 0h base + 44h offset = 44h



STCU2_MBSL field descriptions

Field	Description
0 MBS31	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS30	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS29	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
3 MBS28	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS27	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS26	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS25	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS24	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS23	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS22	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS21	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS20	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS19	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS18	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS17	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS16	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS15	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS14	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS13	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS12	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS11	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS10	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS9	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS8	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS7	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS6	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
26 MBS5	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS4	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS3	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS2	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS1	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS0	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

77.9.18 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63	MBS62	MBS61	MBS60	MBS59	MBS58	MBS57	MBS56	MBS55	MBS54	MBS53	MBS52	MBS51	MBS50	MBS49	MBS48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSM field descriptions

Field	Description
0 MBS63	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS62	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS61	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
3 MBS60	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS59	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS58	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS57	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS56	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS55	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS54	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS53	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS52	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS51	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS50	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS49	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS48	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS47	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS46	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS45	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

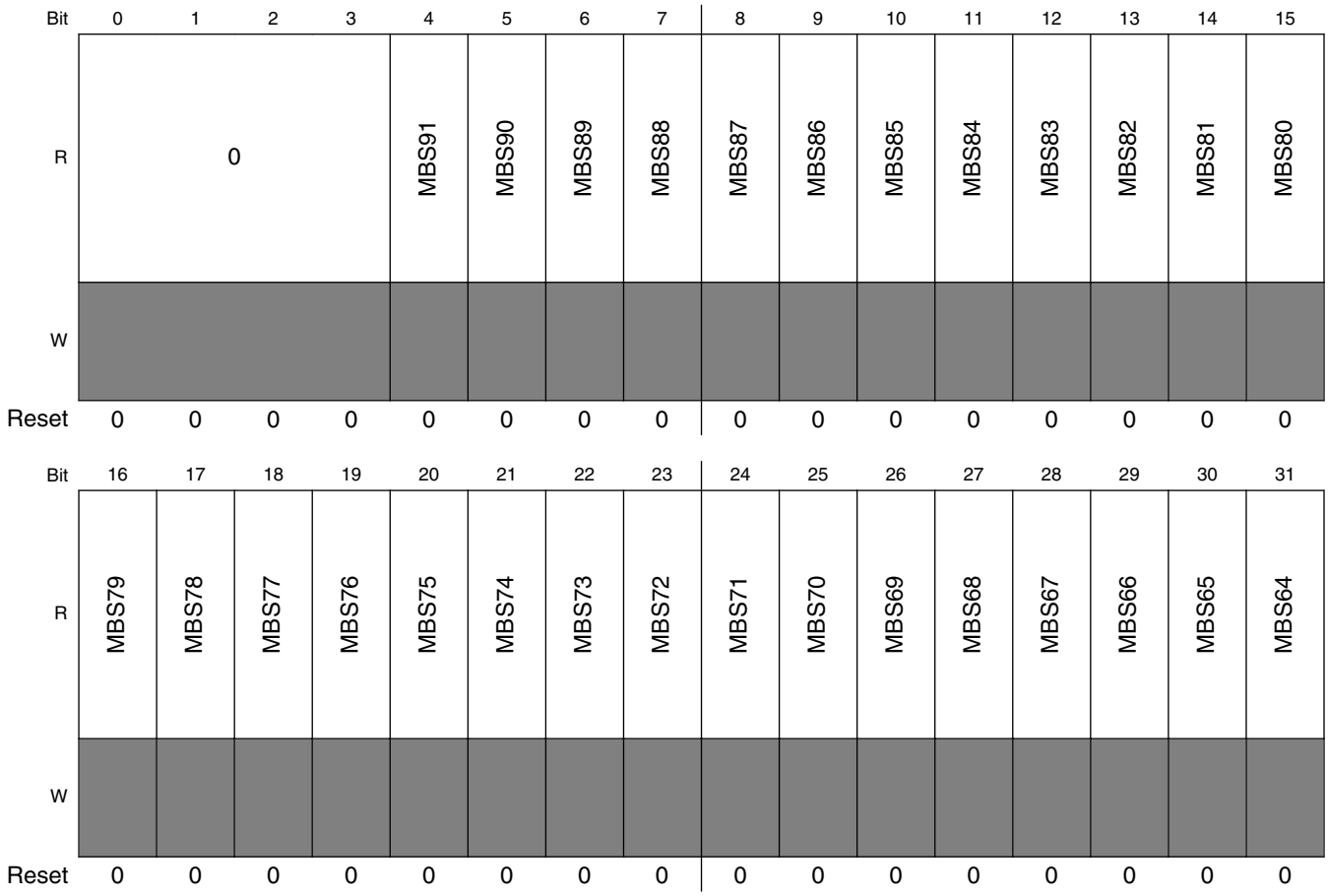
77.9.19 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)

The STCU2_MBSH register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Register description

Address: 0h base + 4Ch offset = 4Ch



STCU2_MBSH field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MBS91	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS90	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS89	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS88	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS87	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS86	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS85	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS84	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS83	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS82	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS81	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS80	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS79	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS78	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS77	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS76	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS75	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS74	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
22 MBS73	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS72	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS71	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS70	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS69	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS68	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS67	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS66	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSH field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS65	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS64	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

77.9.20 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)

The STCU2_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEL field descriptions

Field	Description
0 MBE31	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE30	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBE29	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE28	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE27	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE26	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE23	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE17	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE14	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE11	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

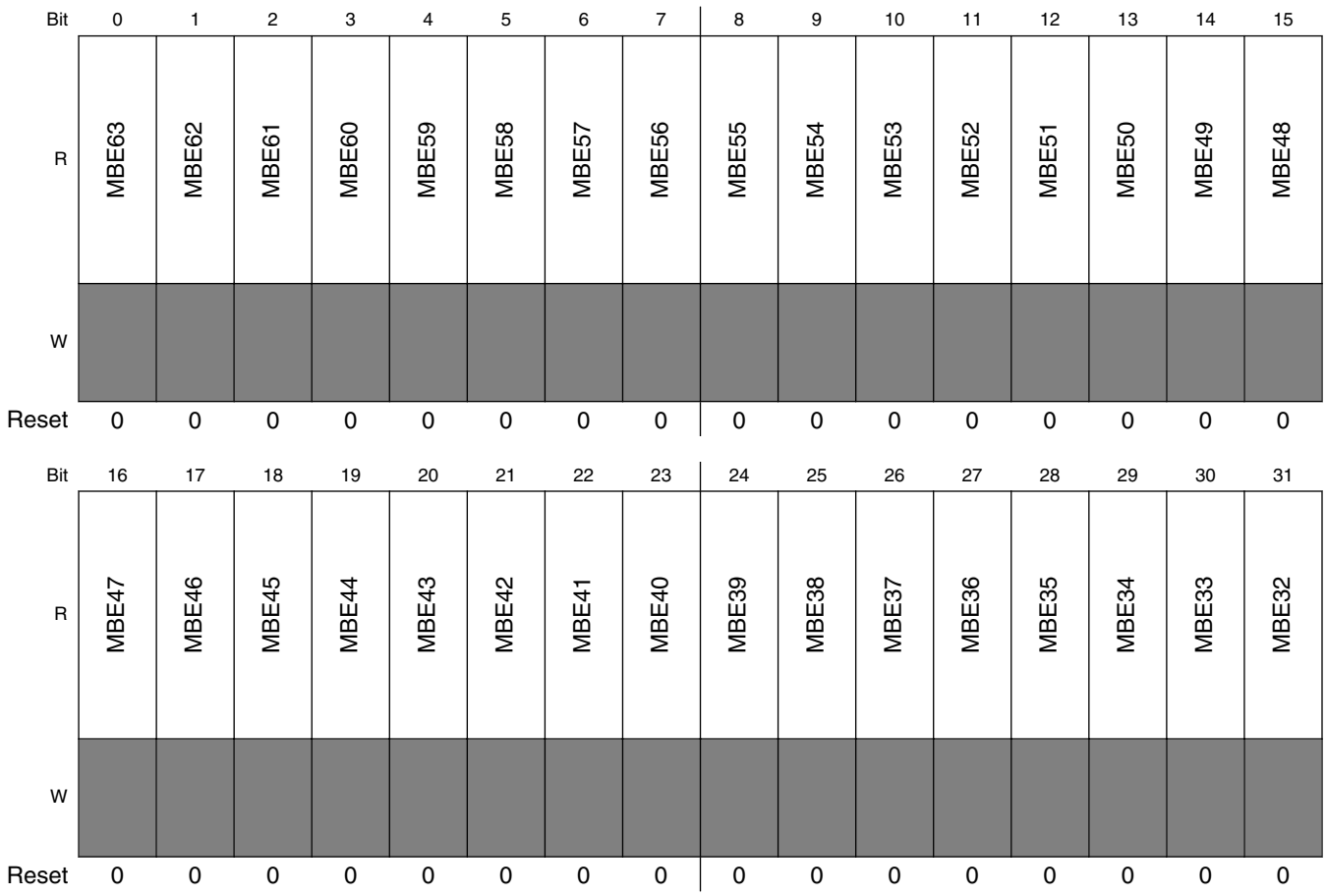
Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE5	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE2	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

77.9.21 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)

The STCU2_MBEM register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 54h offset = 54h



STCU2_MBEM field descriptions

Field	Description
0 MBE63	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE62	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
2 MBE61	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE60	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE59	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE58	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE57	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE56	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE55	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE54	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE53	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE52	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE51	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE50	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
14 MBE49	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE48	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE47	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE46	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE45	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE44	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE43	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE42	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE41	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE40	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE39	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE38	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
26 MBE37	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE36	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE35	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE34	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE33	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE32	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

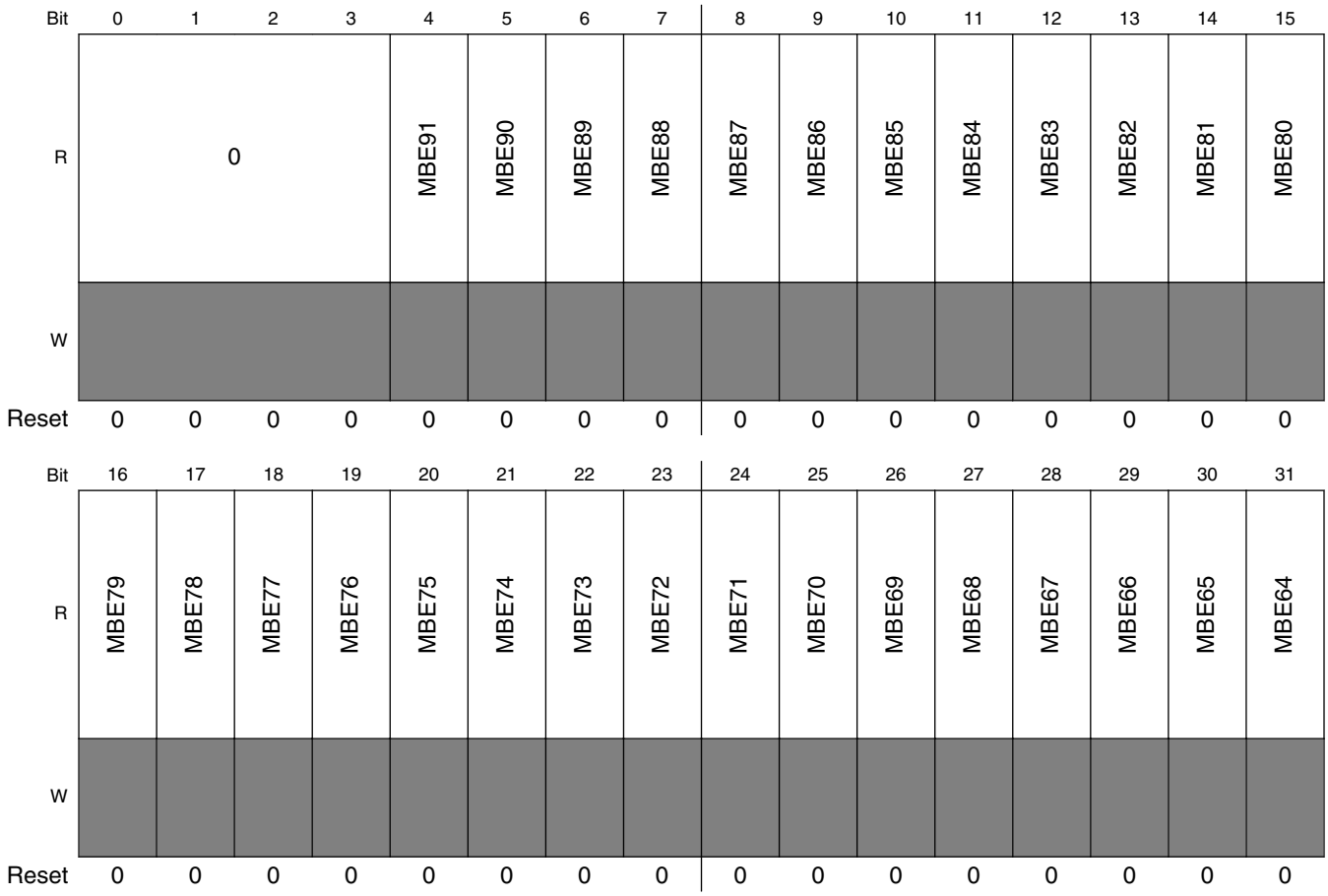
77.9.22 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)

The STCU2_MBEH register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Register description

Address: 0h base + 58h offset = 58h



STCU2_MBEH field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MBE91	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE90	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE89	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE88	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEH field descriptions (continued)

Field	Description
8 MBE87	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE86	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE85	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE84	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE83	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE82	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE81	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE80	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE79	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE78	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE77	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE76	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEH field descriptions (continued)

Field	Description
20 MBE75	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE74	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE73	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE72	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE71	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE70	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE69	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE68	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE67	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE66	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE65	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE64	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

77.9.23 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)

The STCU2_MBSLSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSLSW field descriptions

Field	Description
0 MBSSW31	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW30	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW29	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW28	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW27	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW26	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW25	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW24	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW23	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW22	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW21	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW20	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW19	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW18	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW17	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW16	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW15	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW14	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW13	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW12	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW11	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW10	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW9	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW8	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW7	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW6	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW5	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW4	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW3	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW2	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW1	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW0	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

77.9.24 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)

The STCU2_MBSMSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSMSW field descriptions

Field	Description
0 MBSSW63	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW62	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW61	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW60	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW59	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW58	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW57	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW56	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW55	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW54	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW53	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW52	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW51	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW50	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW49	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW48	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW47	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW46	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW45	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW44	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW43	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW42	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW41	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW40	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW39	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW38	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW37	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW36	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW35	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW34	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW33	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW32	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

77.9.25 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)

The STCU2_MBSHSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				MBSSW91	MBSSW90	MBSSW89	MBSSW88	MBSSW87	MBSSW86	MBSSW85	MBSSW84	MBSSW83	MBSSW82	MBSSW81	MBSSW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW79	MBSSW78	MBSSW77	MBSSW76	MBSSW75	MBSSW74	MBSSW73	MBSSW72	MBSSW71	MBSSW70	MBSSW69	MBSSW68	MBSSW67	MBSSW66	MBSSW65	MBSSW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSHSW field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MBSSW91	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW90	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW89	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW88	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW87	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW86	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW85	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW84	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBSHSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW83	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW82	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW81	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW80	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW79	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW78	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW77	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW76	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW75	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW74	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW73	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW72	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBSHSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW71	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW70	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW69	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW68	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW67	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW66	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW65	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW64	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

77.9.26 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)

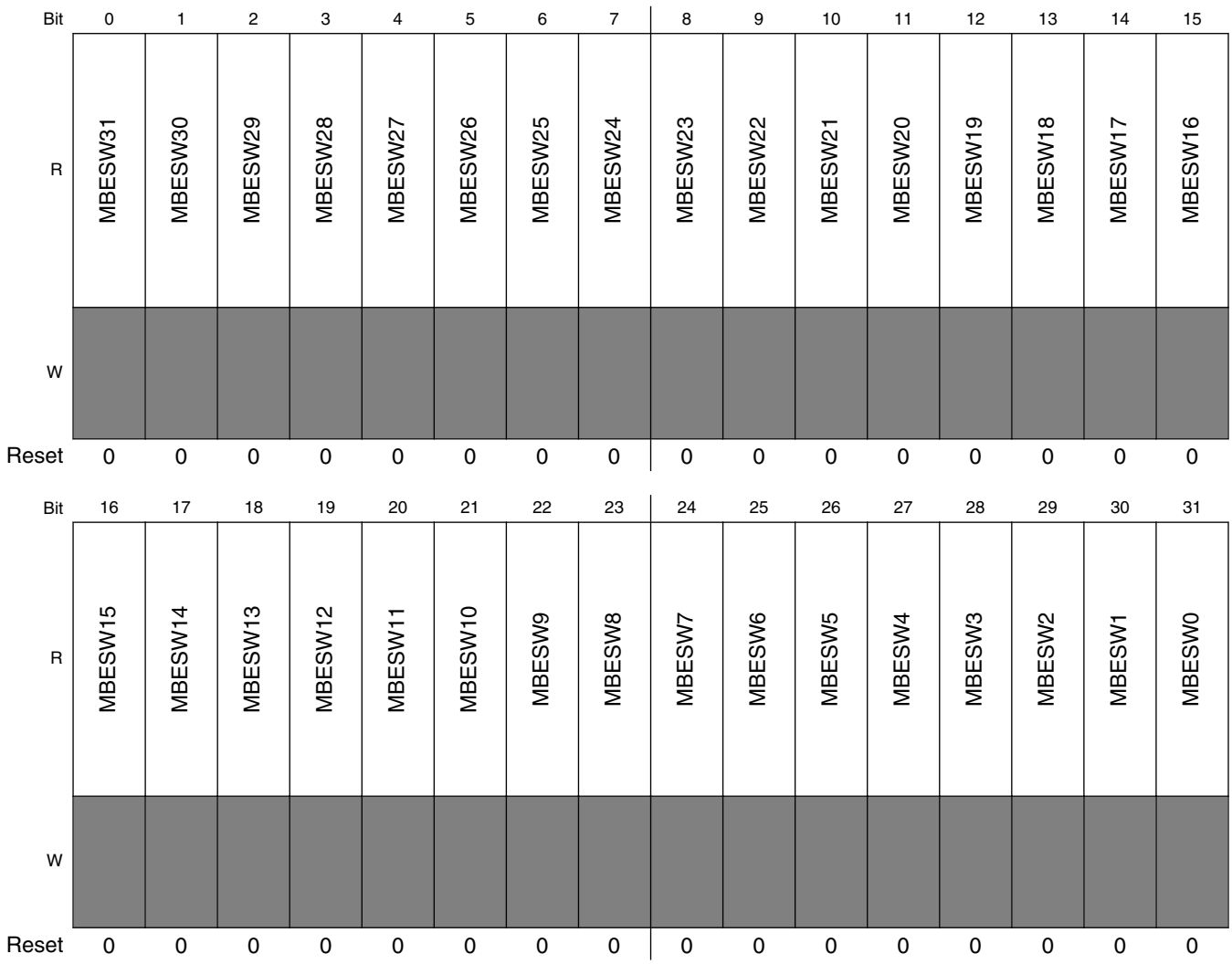
The STCU2_MBELSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 68h offset = 68h



STCU2_MBELSW field descriptions

Field	Description
0 MBESW31	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW30	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW29	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW28	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW27	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW26	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW24	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW23	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW12	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW11	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW0	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

77.9.27 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)

The STCU2_MBEMSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63	MBESW62	MBESW61	MBESW60	MBESW59	MBESW58	MBESW57	MBESW56	MBESW55	MBESW54	MBESW53	MBESW52	MBESW51	MBESW50	MBESW49	MBESW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEMSW field descriptions

Field	Description
0 MBESW63	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW62	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW61	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW60	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW59	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW58	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW57	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW56	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
8 MBESW55	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW54	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW53	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW52	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW51	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW50	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW49	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW48	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW47	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW46	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW45	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW44	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
20 MBESW43	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW42	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW41	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW40	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW39	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW38	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW37	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW36	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW35	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW34	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW33	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW32	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

77.9.28 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)

The STCU2_MBEHSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				MBESW91	MBESW90	MBESW89	MBESW88	MBESW87	MBESW86	MBESW85	MBESW84	MBESW83	MBESW82	MBESW81	MBESW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW79	MBESW78	MBESW77	MBESW76	MBESW75	MBESW74	MBESW73	MBESW72	MBESW71	MBESW70	MBESW69	MBESW68	MBESW67	MBESW66	MBESW65	MBESW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEHSW field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MBESW91	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW90	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW89	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW88	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW87	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW86	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW85	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW84	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW83	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW82	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW81	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW80	MBESWx: On-Line End status (64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBEHSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW79	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW78	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW77	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW76	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW75	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW74	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW73	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW72	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW71	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW70	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW69	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW68	MBESWx: On-Line End status (64 to 95) of the selected MBIST

Table continues on the next page...

STCU2_MBEHSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW67	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW66	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW65	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW64	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

77.9.29 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFML field descriptions

Field	Description
0 MBUFM31	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM30	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM29	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM28	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM27	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM26	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM25	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM24	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM23	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
9 MBUFM22	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM21	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM20	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM19	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM18	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM17	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM16	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM15	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM14	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM13	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM12	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM11	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
21 MBUFM10	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM9	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM8	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM7	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM6	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM5	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM4	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM3	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM2	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM1	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM0	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

77.9.30 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMM field descriptions

Field	Description
0 MBUFM63	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM62	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM61	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM60	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM59	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM58	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM57	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM56	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM55	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM54	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM53	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM52	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM51	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM50	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM49	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM48	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM47	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM46	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM45	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM44	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM43	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM42	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM41	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM40	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM39	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM38	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM37	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM36	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM35	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM34	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM33	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM32	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

77.9.31 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Register description

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				MBUFM91	MBUFM90	MBUFM89	MBUFM88	MBUFM87	MBUFM86	MBUFM85	MBUFM84	MBUFM83	MBUFM82	MBUFM81	MBUFM80
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM79	MBUFM78	MBUFM77	MBUFM76	MBUFM75	MBUFM74	MBUFM73	MBUFM72	MBUFM71	MBUFM70	MBUFM69	MBUFM68	MBUFM67	MBUFM66	MBUFM65	MBUFM64
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMH field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MBUFM91	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM90	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM89	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM88	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM87	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM86	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM85	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMH field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM84	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM83	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM82	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM81	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM80	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM79	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM78	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM77	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM76	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM75	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM74	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM73	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMH field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM72	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM71	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM70	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM69	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM68	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM67	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM66	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM65	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM64	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

77.9.32 STCU2 LBIST Control (STCU2_LB_CTRL*n*)

NOTE

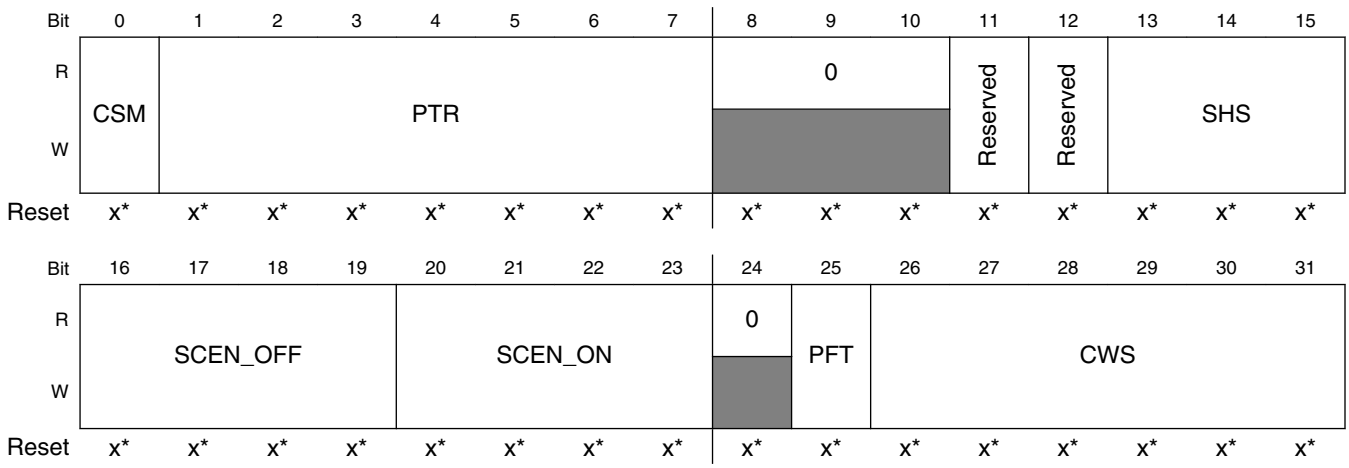
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self test phase is active
- Online self test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 100h offset + (64d × i), where i=0d to 7d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise, it is scheduled sequentially to the completion of the current LBIST execution. 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST pointer PTR defines the logical pointer to the next LBIST. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current LBIST has been completed. LBIST POINTERS 0000b - pointer to LBIST partition 0 0001b - pointer to LBIST partition 1 0010b - pointer to LBIST partition 2 0011b - pointer to LBIST partition 3 0100b - pointer to LBIST partition 4 0101b - pointer to LBIST partition 5 0110b - pointer to LBIST partition 6 0111b - pointer to LBIST partition 7

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	... 0x7F - pointer to NIL, end of LBIST execution Values not defined in this list will result in an error being set in the STCU2_ERR register.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. **NOTE:** This reserved field is writeable but do not write any value to it other than its reset value
12 Reserved	This field is reserved. **NOTE:** This reserved field is writeable but do not write any value to it other than its reset value
13–15 SHS	Shift speed SHS defines the shift speed 000 Shift at full rate of STCU2 core clock 001 Shift at 1/2 rate of STCU2 core clock 010 Shift at 1/3 rate of STCU2 core clock 011 Shift at 1/4 rate of STCU2 core clock 100 Shift at 1/5 rate of STCU2 core clock 101 Shift at 1/6 rate of STCU2 core clock 110 Shift at 1/7 rate of STCU2 core clock 111 Shift at 1/8 rate of STCU2 core clock
16–19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN **NOTE:** Scen_off must be programmed to a value >=1 0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
20–23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN **NOTE:** Scen_on delay register value must be programmed to a value >=1>

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 PFT	Past Flush Test The LBIST controller, by default, applies 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence.
26–31 CWS	Capture window size CWS defines the capture window size. 000000: Illegal 000001: Controller waits 1 shift cycle for capture to finish. 000010: Controller waits 2 shift cycles for capture to finish. 000011: Controller waits 3 shift cycles for capture to finish. ... 111110: Controller waits 62 shift cycles for capture to finish. 111111: Controller waits 63 shift cycles for capture to finish.

77.9.33 STCU2 LBIST PC Stop Register (STCU2_LB_PCS*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 104h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PCS																									
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PCS*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

77.9.34 STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREL register defines the LSB part (31-0) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 110h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISREx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREL*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

77.9.35 STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREH register defines the MSB part (63-32) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 114h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISREx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREH*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

77.9.36 STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRL register reports the LSB part (31-0) of the MISR obtained at the end of the Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 118h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRL*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read Low Bits This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution

77.9.37 STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRH register report the MSB part (63-32) of the MISR obtained at the end of each Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 11Ch offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRH*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read High Bits This field is equivalent to the High Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution

77.9.38 STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRELSW register defines the LSB part (31-0) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 120h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRELSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

77.9.39 STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREHSW register defines the MSB part (63-32) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 124h offset + (64d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREHSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

77.9.40 STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW*n*)

NOTE

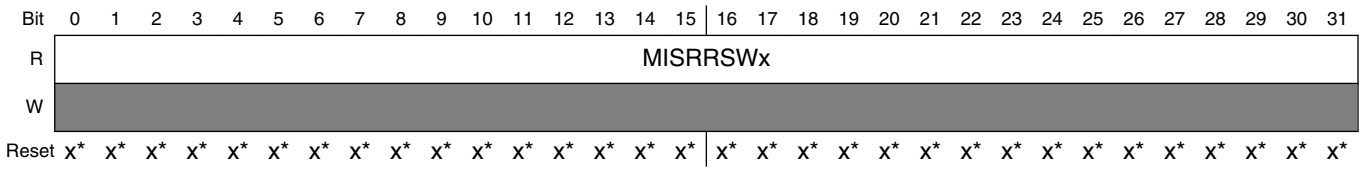
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRLSW register report the LSB part (31-0) of the MISR obtained at the end of the On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 128h offset + (64d × i), where i=0d to 7d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRLSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read Low Bin This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution.

77.9.41 STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW*n*)

NOTE

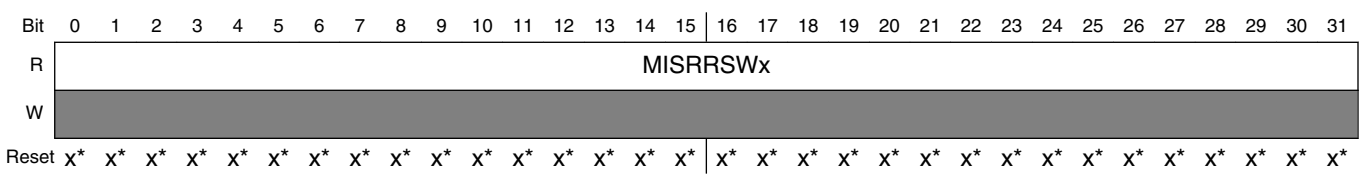
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRHSW register report the MSB part (63-32) of the MISR obtained at the end of each On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 12Ch offset + (64d × i), where i=0d to 7d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRHSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read High Bits This field is equivalent to the Low Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution.

77.9.42 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

The offset of this register is a function of NMCUT .

Address: 0h base + 600h offset + (4d × i), where i=0d to 91d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM	PTR							0							
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST or MBIST pointer

Table continues on the next page...

STCU2_MB_CTRL*n* field descriptions (continued)

Field	Description
	PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST orMBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure is stopped when the current MBIST has been completed. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU2_ERR register.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.10 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

77.10.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and L/MBIST execution parameters stored into the NVM memory.The target is to cover the amount of physical defects in the digital logic and in the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU2_LBUFM register
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU2_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU2_ERR_FM register
- Program the STCU2_LB_CTRL, STCU2_LB_PCS, STCU2_MISREL/H registers of each LBIST to be executed.
- Program the STCU2_MB_CTRL registers of each NMCUT to be executed

- Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution
- Program the STCU2_CFG register to define: the core and LBIST/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed
- In case the internal CRC comparison has been enabled, program the CRCE value expected at the end of the off-line Self Test sequence into the STCU2_CRCE register
- In case the PLL usage has been enabled, program the STCU2_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU2_RUN register to enable the Off-Line Self-Test execution and the L/MBPLLEN bits to enable the PLL during L/MBIST execution according to STCU2_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and L/MBIST execution order, when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register.
- The STCU2 switches off the core clock at the end of the Self-Test run and releases the PLL control in case this feature has been enabled.
- The STCU2 resets the STCU2_RUN[RUN] bit and the device exits from boot sequence. In case there are failures, the STCU2 flags these failures toward the FCCU.
- The software application reads the STCU2_LBS flag register to check the failed LBIST when UF or RF conditions is/are detected.
- The software application reads the STCU2_LBE flag register to check the LBIST still on-going when UF or RF is/are detected.
- The software application reads the STCU2_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application reads the STCU2_MBEL/M/H flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
- The software application reads the bits INV, ENGE, CRCS, WDTO, LOCKE of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.

- In case the CRCEN bit has been enabled, the software application read the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU2_ERR register.
- The software application reads for each device's LBIST, the STCU2_LBMISREL/H and STCU2_LBIST_MISRRL/H registers to check the coherence with the STCU2_LBS bits.

NOTE

The software application reads operations on the STCU2 CRC and LBIST MISR registers after off-line self test execution has been successfully performed are just an additional reliability layer added to improve the already implemented Auto Self-Test feature included into STCU2.

77.10.2 Online self test sequence

The online self test sequence is the most commonly used STCU2 application mode. The IPS interface is used to program the STCU2 registers and to schedule LBIST and MBIST execution. Since online LBIST and MBIST operations clean-up the content of Digital Logic (LBIST) or RAM (MBIST) tested, activation of this mode must be approached with caution. The suggested sequence is the following:

1. Ensure the RUN_CFG field value in all implemented MC_ME_PCTL*n* registers forces the associated peripheral to a frozen, clock gated state.
2. Write the Key1/Key2 sequence to the SKC register to unlock online STCU2 access.
3. Ensure CFG[WRP]=0 to enable IPS bus access to the online self test registers.
4. If required, program the following registers to select the LBIST, MBIST and STCU2 internal fault reaction: LBUFM, MBUFML/M/H and ERR_FM.

NOTE

The LBIST, MBIST and STCU2 internal fault reaction should have been already programmed during the offline self test sequence and so there should be no need to program again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten.

5. Select the reset management options for online LBIST execution by programming the LBRMSW register.
6. The LB_CTRL and LB_PCS registers should have been already programmed during offline self test sequence and so they should not need to be programmed again.

However, if they have not been programmed or there is the need of changing them for any reason, they can be overwritten. The registers LB_MISRELSW/HSW have to be programmed according to the expected signature.

7. Overwrite/Program the MB_CTRL registers of each NMCUT to be executed.
8. Overwrite/Program the WDG register to define the time budget assigned for the L/MBIST execution
9. Overwrite/Program the CFG register to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the CRCREN bit to enable the self-checking feature and finally set the pointer to the first LBIST/NMCUT to be executed.
10. In case the internal CRC comparison has been enabled, overwrite/program the CRCE value expected at the end of the online self test sequence into the CRCE register
11. Program the RUNSW[RUNSW] field to enable online self test execution; the L/MBSWPLLEN bits to enable the PLL lock signal monitor during L/MBIST execution
12. Start the online self test and depending on PLL management and L/MBIST execution order, also when PLL LOCK is high.
13. Wait the WDGEOC run time execution specified into the WDG register
14. The STCU2 switch-off the core clock at the end of the self test run
15. The functional reset activated is managed by the LBRMSW register (See step 5).
16. In case, at least one of the selected LBIST enables the Global functional reset, the STCU2 requests a functional reset which restarts the system while, when all the selected LBIST enables its own dedicated functional reset, only these local reset lines are applied and only the related LBIST partitions are cleaned-up.
17. The STCU2 resets the RUNSW[RUNSW] field. In both the reset cases and in case of fault/s, the STCU2 flags the failure toward the FCCU.
18. The software application reads the LBSSW flag register to check the failed LBIST when UF or RF conditions is/are detected.
19. The software application reads the LBESW flag register to check if the LBIST is still on-going when UF or RF is/are detected.
20. The software application reads the MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
21. The software application reads the MBELSW/MSW/HSW flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
22. The software application reads the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.

23. In case the CRCEN bit has been enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCSSW of the STCU2_ERR register.
24. The software application reads for each device's LBIST, the LBMISRELSW/HSW and LBIST_MISRRLSW/HSW registers to check the coherence with the LBS bits

NOTE

1. A reset is necessary after running any kind of online self test.
2. The software application read operations on the STCU2 CRC and LBIST MISR registers after successful online self test execution are an additional reliability layer to improve the auto-self-test feature included in the STCU2.
3. Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.

77.10.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for applications requiring improved reliability. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU2_SKC register
2. Set the BYP bit in the STCU2_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

77.10.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT BIST can be run once during the online or offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- Because software has full control of the system while applications are running during the online self-test, STCU2 is not allowed to take control of the PLL but can just monitor the lock signal to flag wrong unlock events.

Chapter 78
Error Injection Module (EIM)

78.1 Chip-specific EIM information

The EIM has only one channel (channel 0). It is used to inject errors on the DMA TCD RAM (64-bit data, 8-bit ECC) interface.

78.2 Introduction

The Error Injection module is mainly used for diagnostic purposes. It provides a method for diagnostic coverage of the peripheral memories. See the chip-specific EIM information to determine which peripheral memories are supported by this method.

78.2.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

NOTE

The following diagram shows an example EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

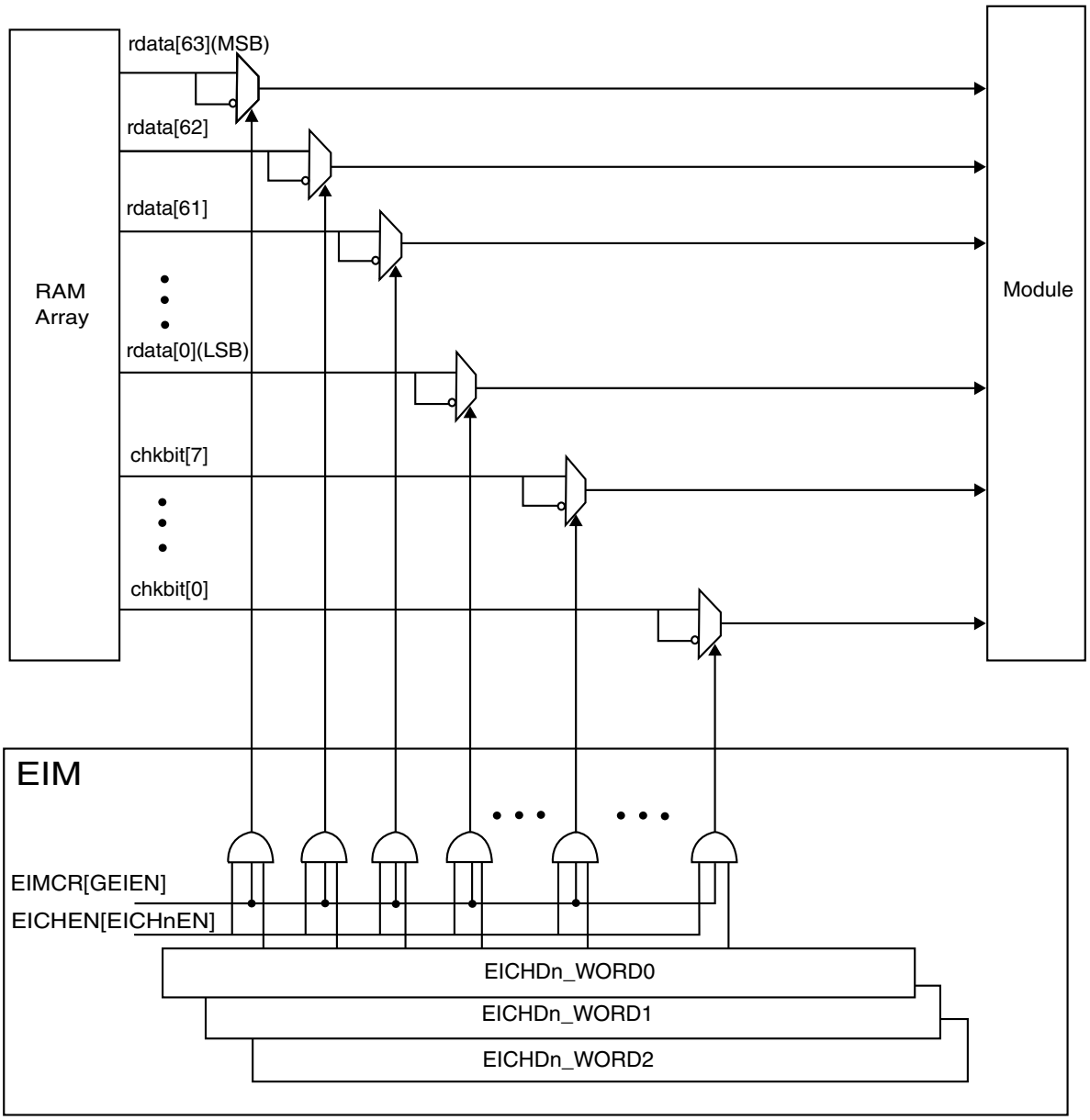


Figure 78-1. EIM functional block diagram (64-bit read data bus and 8-bit checkbit bus)

78.2.2 Features

The EIM includes these features:

- Supports 1 error injection channel
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

78.3 Memory map and register definition

The EIM provides an IPS programming model mapped to an on-platform peripheral slot.

Programming model access

All system bus masters can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an IPS error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation will result in non-deterministic behavior.

Error injection channel descriptor: function and structure

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and checkbit bus from target RAM are inverted on a read access.
- Consists of a 128-bit (16-byte) structure, composed of four 32-bit words, in the EIM programming model.
  - Each descriptor consists of Error Injection Channel Descriptor Word0-1 (EICHD*n*_WORD0-1) and, depending on the implementation, Word2 (EICHD*n*_WORD2).
  - When Word2 is unused, the corresponding word in the 16-byte structure is unused.
  - The fourth word in the 16-byte structure is always unused.

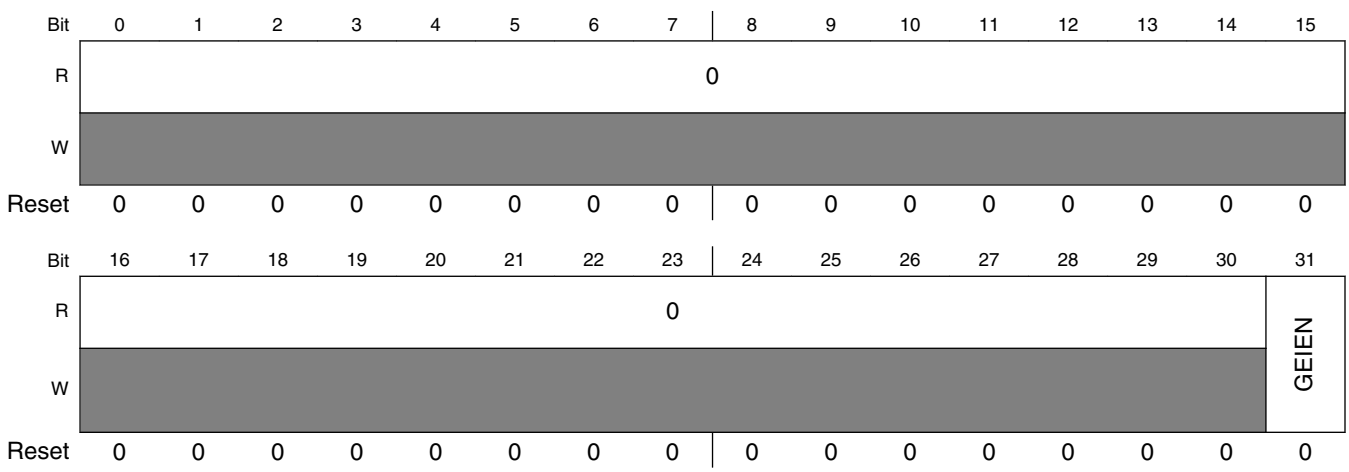
EIM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Error Injection Module Configuration Register (EIM_EIMCR)	32	R/W	0000_0000h	78.3.1/3898
4	Error Injection Channel Enable register (EIM_EICHEN)	32	R/W	0000_0000h	78.3.2/3899
100	Error Injection Channel Descriptor, Word0 (EIM_EICHD0_WORD0)	32	R/W	0000_0000h	78.3.3/3900
104	Error Injection Channel Descriptor, Word1 (EIM_EICHD0_WORD1)	32	R/W	0000_0000h	78.3.4/3901
108	Error Injection Channel Descriptor, Word2 (EIM_EICHD0_WORD2)	32	R/W	0000_0000h	78.3.5/3901

78.3.1 Error Injection Module Configuration Register (EIM_EIMCR)

The EIM Configuration Register is used to globally enable/disable the error injection function.

Address: 0h base + 0h offset = 0h



EIM_EIMCR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GEIEN	Global Error Injection Enable This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0 Disabled 1 Enabled

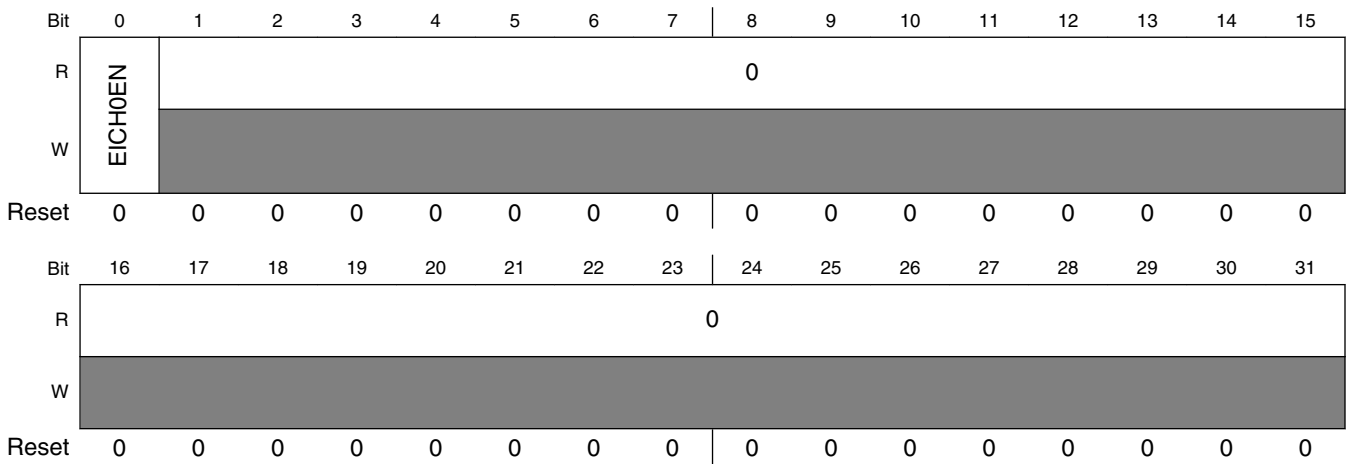
78.3.2 Error Injection Channel Enable register (EIM_EICHEN)

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

NOTE

To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

Address: 0h base + 4h offset = 4h



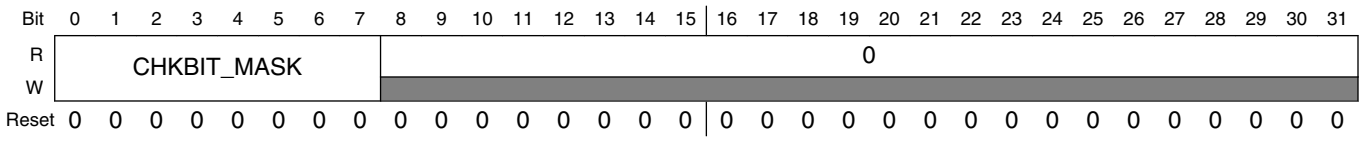
EIM_EICHEN field descriptions

Field	Description
0 EICH0EN	Error Injection Channel 0 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 0 1 Error injection is enabled on Error Injection Channel 0
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

78.3.3 Error Injection Channel Descriptor, Word0 (EIM_EICHD*n*_WORD0)

The first word of the Error Injection Channel Descriptor defines a left-justified mask field (CHKBIT_MASK) whose width is up to 8 bits. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Address: 0h base + 100h offset + (256d × i), where i=0d to 0d



EIM_EICHD*n*_WORD0 field descriptions

Field	Description
0–7 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. The width of the field for the channel is defined in the chip-specific EIM information. 0 The corresponding bit of the checkbit bus remains unmodified. 1 The corresponding bit of the checkbit bus is inverted.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

78.3.4 Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1)

The second word of the Error Injection Channel Descriptor defines the bits in the mask corresponding to bytes 0–3 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 104h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B0_3DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD1 field descriptions

Field	Description
0–31 B0_3DATA_ MASK	Data Mask Bytes 0-3 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 0-3 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 0-3 on the read data bus remains unmodified 1 The corresponding bit of bytes 0-3 on the read data bus is inverted.

78.3.5 Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)

The third word of the Error Injection Channel Descriptor, when present, defines the bits in the mask corresponding to bytes 4–7 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 108h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B4_7DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD2 field descriptions

Field	Description
0–31 B4_7DATA_ MASK	Data Mask Bytes 4-7 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 4-7 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 4-7 on the read data bus remains unmodified. 1 The corresponding bit of bytes 4-7 on the read data bus is inverted.

78.4 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel:

- Write 1 to the EICHEN[EICH*n*EN] field, where *n* denotes the channel number.
- Write 1 to EIMCR[GEIEN].

NOTE

When the use case for a channel requires writing any EICHD*n*_WORD register, write the EICHD*n*_WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD*n*_WORD register clears the corresponding EICHEN[EICH*n*EN] field.

The EIM supports 1 error injection channel. Each channel:

- is assigned to a single memory array interface.
- intercepts the assigned memory read data bus and checkbit bus and injects errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD*n*_WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 78-1 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

78.4.1 Error injection scenarios

The EIM supports these cases of error injection:

- To generate a single-bit error, invert only 1 bit of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.
- To generate a multi-bit error, invert only 2 bits of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.

NOTE

An attempt to invert more than 2 bits in one operation might result in undefined behavior.

Chapter 79
Register Protection (REG_PROT)

79.1 Chip-specific REG PROT information

79.1.1 Register protection (REG_PROT) configuration

The following table lists the all of the protected registers.

Table 79-1. Protected registers

Module	Register	Reg offset	Protect size[1]	Description
ADC_0	MCR	0x00	3	Main Configuration Register
ADC_0	IMR	0x20	3	Interrupt Mask Register
ADC_0	CIMR0	0x24	3	Channel Interrupt Mask Register
ADC_0	WTIMR	0x34	3	Watchdog Interrupt Threshold Mask Reg
ADC_0	DMAE	0x40	3	DMA Enable Register
ADC_0	DMAR0	0x44	3	DMA Channel Select Register 0
ADC_0	THRHLR0	0x60	3	Threshold Register 0
ADC_0	THRHLR1	0x64	3	Threshold Register 1
ADC_0	THRHLR2	0x68	3	Threshold Register 2
ADC_0	THRHLR3	0x6C	3	Threshold Register 3
ADC_0	PSCR	0x80	3	PRESAMPLING CONTROL REGISTER
ADC_0	PSR0	0x84	3	PRESAMPLING REGISTER
ADC_0	CTR0	0x94	3	Conversion Timing Register 0
ADC_0	NCMR0	0xA4	3	Normal Conversion Mask Register 0
ADC_0	JCMR0	0xB4	3	Injected Conversion Mask Register 0
ADC_0	PDEDR	0xC8	3	Power Down Exit Delay Register
ADC_0	THRHLR4	0x280	3	Threshold Register 4
ADC_0	THRHLR5	0x284	3	Threshold Register 5
ADC_0	THRHLR6	0x288	3	Threshold Register 6
ADC_0	THRHLR7	0x28C	3	Threshold Register 7
ADC_0	THRHLR8	0x290	3	Threshold Register 8

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ADC_0	THRHLR9	0x294	3	Threshold Register 9
ADC_0	THRHLR10	0x298	3	Threshold Register 10
ADC_0	THRHLR11	0x29C	3	Threshold Register 11
ADC_0	THRHLR12	0x2A0	3	Threshold Register 12
ADC_0	THRHLR13	0x2A4	3	Threshold Register 13
ADC_0	THRHLR14	0x2A8	3	Threshold Register 14
ADC_0	THRHLR15	0x2AC	3	Threshold Register 15
ADC_0	CWSELR0	0x2B0	3	WATCHDOG SELECT REGISTER
ADC_0	CWSELR1	0x2B4	3	WATCHDOG SELECT REGISTER
ADC_0	CWENR0	0x2E0	3	CChannel watchdog enable register
ADC_0	AWORR0	0x2F0	3	ANALOG WATCHDOG OUT of RANGE REGISTER
ADC_0	STCR1	0x340	3	SELF TEST CONFIGURATION REGISTER 1
ADC_0	STCR2	0x344	3	SELF TEST CONFIGURATION REGISTER 2
ADC_0	STCR3	0x348	3	SELF TEST CONFIGURATION REGISTER 3
ADC_0	STBRR	0x34C	3	SELF TEST BAUD RATE REGISTER
ADC_0	STAW0R	0x380	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	STAW1AR	0x384	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	STAW1BR	0x388	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	STAW2R	0x38C	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	STAW4R	0x394	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	STAW5R	0x398	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_0	CALBISTREG	0x3A0	3	Calibration BIST register
ADC_0	OFSGNUSR	0x3A8	3	Offset gain user
ADC_1	MCR	0x00	3	Main Configuration Register
ADC_1	IMR	0x20	3	Interrupt Mask Register
ADC_1	CIMR0	0x24	3	Channel Interrupt Mask Register
ADC_1	WTIMR	0x34	3	Watchdog Interrupt Threshold Mask Reg
ADC_1	DMAE	0x40	3	DMA Enable Register
ADC_1	DMAR0	0x44	3	DMA Channel Select Register 0
ADC_1	THRHLR0	0x60	3	Threshold Register 0
ADC_1	THRHLR1	0x64	3	Threshold Register 1
ADC_1	THRHLR2	0x68	3	Threshold Register 2

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ADC_1	THRHLR3	0x6C	3	Threshold Register 3
ADC_1	PSCR	0x80	3	PRESAMPLING CONTROL REGISTER
ADC_1	PSR0	0x84	3	PRESAMPLING REGISTER
ADC_1	CTR0	0x94	3	Conversion Timing Register 0
ADC_1	NCMR0	0xA4	3	Normal Conversion Mask Register 0
ADC_1	JCMR0	0xB4	3	Injected Conversion Mask Register 0
ADC_1	PDEDR	0xC8	3	Power Down Exit Delay Register
ADC_1	THRHLR4	0x280	3	Threshold Register
ADC_1	THRHLR5	0x284	3	Threshold Register
ADC_1	THRHLR6	0x288	3	Threshold Register
ADC_1	THRHLR7	0x28C	3	Threshold Register
ADC_1	THRHLR8	0x290	3	Threshold Register
ADC_1	THRHLR9	0x294	3	Threshold Register
ADC_1	THRHLR10	0x298	3	Threshold Register
ADC_1	THRHLR11	0x29C	3	Threshold Register
ADC_1	THRHLR12	0x2A0	3	Threshold Register
ADC_1	THRHLR13	0x2A4	3	Threshold Register
ADC_1	THRHLR14	0x2A8	3	Threshold Register
ADC_1	THRHLR15	0x2AC	3	Threshold Register
ADC_1	CWSELR0	0x2B0	3	WATCHDOG SELECT REGISTER
ADC_1	CWSELR1	0x2B4	3	WATCHDOG SELECT REGISTER
ADC_1	CWENR0	0x2E0	3	channel watchdog enable register
ADC_1	AWORR0	0x2F0	3	ANALOG WATCHDOG OUT of RANGE REGISTER
ADC_1	STCR1	0x340	3	SELF TEST CONFIGURATION REGISTER 1
ADC_1	STCR2	0x344	3	SELF TEST CONFIGURATION REGISTER 2
ADC_1	STCR3	0x348	3	SELF TEST CONFIGURATION REGISTER 3
ADC_1	STBRR	0x34C	3	SELF TEST BAUD RATE REGISTER
ADC_1	STAW0R	0x380	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_1	STAW1AR	0x384	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_1	STAW1BR	0x388	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_1	STAW2R	0x38C	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_1	STAW4R	0x394	3	SELF TEST ANALOG WATCHDOG REGISTER

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ADC_1	STAW5R	0x398	3	SELF TEST ANALOG WATCHDOG REGISTER
ADC_1	CALBISTREG	0x3A0	3	Calibration BIST register
ADC_1	OFSGNUSR	0x3A8	3	Offset gain user
AFE	OSCCTRL	0x00	3	Oscillator Control Register
AFE	OSCSTS	0x04	3	Oscillator Status Register
AFE	OSCDLY	0x08	3	Oscillator Delay Register
AFE	PLLCTRL1	0x0C	3	SDPLL Control Register 1
AFE	PLLCTRL2	0x10	3	SDPLL Control Register 2
AFE	PLLCTRL3	0x14	3	SDPLL Control Register 3
AFE	PLLCTRL8	0x28	3	SDPLL Control Register 8
AFE	PLLSTS	0x2c	3	SDPLL Status Register
AFE	ADCCTRL1	0x30	3	ADC Control Register 1
AFE	ADCCTRL2	0x34	3	ADC Control Register 2
AFE	ADCTOT	0x38	3	ADC Tracking Oscillator Trim Register
AFE	ADCRST	0x3c	3	ADC Reset Register
AFE	ADCTRIM	0x40	3	ADC Trim Register
AFE	ADCCTRL7	0x48	3	ADC Control Register 7
AFE	ADCOVLD	0x4c	3	ADC Overload Detect Register
AFE	DACCTRL	0x50	3	DAC Control Register
AFE	VRFCTRL1	0x54	3	VREF Control Register 1
AFE	LVDSTS	0x5c	3	Low Voltage Detect Status Register
AFE	VRGCTRL2	0x64	3	VREG2 Control Register
AFE	VRGCTRL3	0x68	3	VREG3 Control Register
AFE	VRGCTRL4	0x6c	3	VREG4 Control Register
AFE	VRGCTRL5	0x70	3	VREG5 Control Register
AFE	VRGCTRL6	0x74	3	VREG6 Control Register
AFE	VRGCTRL7	0x78	3	VREG7 Control Register
AFE	VRGCTRL8	0x7c	3	VREG8 Control Register
AFE	VRGCTRL9	0x84	3	VREG9 Control Register
AFE	FILTCTRL0	0xa0	3	Decimation Filter Control Register
AFE	FILTCTRL1	0xa4	3	Decimation Filter Control Register
AFE	FILTCTRL2	0xa8	3	Decimation Filter Control Register
AFE	FILTCTRL3	0xac	3	Decimation Filter Control Register
AFE	FLCOEF00	0xc0	3	Decimation Filter Coefficient Register
AFE	FLCOEF10	0xc4	3	Decimation Filter Coefficient Register
AFE	FLCOEF20	0xc8	3	Decimation Filter Coefficient Register
AFE	FLCOEF30	0xcc	3	Decimation Filter Coefficient Register
AFE	FLCOEF40	0xd0	3	Decimation Filter Coefficient Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
AFE	FLCOEF50	0xd4	3	Decimation Filter Coefficient Register
AFE	FLCOEF60	0xd8	3	Decimation Filter Coefficient Register
AFE	FLCOEF70	0xdc	3	Decimation Filter Coefficient Register
AFE	FLCOEF80	0xe0	3	Decimation Filter Coefficient Register
AFE	FLCOEF90	0xe4	3	Decimation Filter Coefficient Register
AFE	FLCOEF100	0xe8	3	Decimation Filter Coefficient Register
AFE	FLCOEF110	0xec	3	Decimation Filter Coefficient Register
AFE	FLCOEF120	0xf0	3	Decimation Filter Coefficient Register
AFE	FLCOEF130	0xf4	3	Decimation Filter Coefficient Register
AFE	FLCOEF140	0xf8	3	Decimation Filter Coefficient Register
AFE	FLCOEF150	0xfc	3	Decimation Filter Coefficient Register
AFE	FLCOEF01	0x100	3	Decimation Filter Coefficient Register
AFE	FLCOEF11	0x104	3	Decimation Filter Coefficient Register
AFE	FLCOEF21	0x108	3	Decimation Filter Coefficient Register
AFE	FLCOEF31	0x10c	3	Decimation Filter Coefficient Register
AFE	FLCOEF41	0x110	3	Decimation Filter Coefficient Register
AFE	FLCOEF51	0x114	3	Decimation Filter Coefficient Register
AFE	FLCOEF61	0x118	3	Decimation Filter Coefficient Register
AFE	FLCOEF71	0x11c	3	Decimation Filter Coefficient Register
AFE	FLCOEF81	0x120	3	Decimation Filter Coefficient Register
AFE	FLCOEF91	0x124	3	Decimation Filter Coefficient Register
AFE	FLCOEF101	0x128	3	Decimation Filter Coefficient Register
AFE	FLCOEF111	0x12c	3	Decimation Filter Coefficient Register
AFE	FLCOEF121	0x130	3	Decimation Filter Coefficient Register
AFE	FLCOEF131	0x134	3	Decimation Filter Coefficient Register
AFE	FLCOEF141	0x138	3	Decimation Filter Coefficient Register
AFE	FLCOEF151	0x13c	3	Decimation Filter Coefficient Register
AFE	FLCOEF02	0x140	3	Decimation Filter Coefficient Register
AFE	FLCOEF12	0x144	3	Decimation Filter Coefficient Register
AFE	FLCOEF22	0x148	3	Decimation Filter Coefficient Register
AFE	FLCOEF32	0x14c	3	Decimation Filter Coefficient Register
AFE	FLCOEF42	0x150	3	Decimation Filter Coefficient Register
AFE	FLCOEF52	0x154	3	Decimation Filter Coefficient Register
AFE	FLCOEF62	0x158	3	Decimation Filter Coefficient Register
AFE	FLCOEF72	0x15c	3	Decimation Filter Coefficient Register
AFE	FLCOEF82	0x160	3	Decimation Filter Coefficient Register
AFE	FLCOEF92	0x164	3	Decimation Filter Coefficient Register
AFE	FLCOEF102	0x168	3	Decimation Filter Coefficient Register
AFE	FLCOEF112	0x16c	3	Decimation Filter Coefficient Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
AFE	FLCOEF122	0x170	3	Decimation Filter Coefficient Register
AFE	FLCOEF132	0x174	3	Decimation Filter Coefficient Register
AFE	FLCOEF142	0x178	3	Decimation Filter Coefficient Register
AFE	FLCOEF152	0x17c	3	Decimation Filter Coefficient Register
AFE	FLCOEF03	0x180	3	Decimation Filter Coefficient Register
AFE	FLCOEF13	0x184	3	Decimation Filter Coefficient Register
AFE	FLCOEF23	0x188	3	Decimation Filter Coefficient Register
AFE	FLCOEF33	0x18c	3	Decimation Filter Coefficient Register
AFE	FLCOEF43	0x190	3	Decimation Filter Coefficient Register
AFE	FLCOEF53	0x194	3	Decimation Filter Coefficient Register
AFE	FLCOEF63	0x198	3	Decimation Filter Coefficient Register
AFE	FLCOEF73	0x19c	3	Decimation Filter Coefficient Register
AFE	FLCOEF83	0x1a0	3	Decimation Filter Coefficient Register
AFE	FLCOEF93	0x1a4	3	Decimation Filter Coefficient Register
AFE	FLCOEF103	0x1a8	3	Decimation Filter Coefficient Register
AFE	FLCOEF113	0x1ac	3	Decimation Filter Coefficient Register
AFE	FLCOEF123	0x1b0	3	Decimation Filter Coefficient Register
AFE	FLCOEF133	0x1b4	3	Decimation Filter Coefficient Register
AFE	FLCOEF143	0x1b8	3	Decimation Filter Coefficient Register
AFE	FLCOEF153	0x1bc	3	Decimation Filter Coefficient Register
CRC_0	CFG1	0x00	3	Configuration Register
CRC_0	CFG2	0x10	3	Configuration Register
CRC_0	CFG3	0x20	3	Configuration Register
CRC_1	CFG1	0x00	3	Configuration Register
CRC_1	CFG2	0x10	3	Configuration Register
CRC_1	CFG3	0x20	3	Configuration Register
CTE	CNTRL	0x00	3	Control Register
CTE	CNTRL1	0x04	3	CTE Control Register 1
CTE	INTEN	0x220	2	CTE Interrupt Enable Register
CTU	TGSISR	0x00	3	Trigger Generator Subunit Input Selection Register
CTU	TGSCR	0x04	2	Trigger Generator Subunit Control Register
CTU	T0CR	0x06	2	Trigger 0 Compare Register
CTU	T1CR	0x08	2	Trigger 1 Compare Register
CTU	T2CR	0x0A	2	Trigger 2 Compare Register
CTU	T3CR	0x0C	2	Trigger 3 Compare Register
CTU	T4CR	0x0E	2	Trigger 4 Compare Register
CTU	T5CR	0x10	2	Trigger 5 Compare Register
CTU	T6CR	0x12	2	Trigger 6 Compare Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CTU	T7CR	0x14	2	Trigger 7 Compare Register
CTU	TGSCCR	0x16	2	TGS Counter Compare Register
CTU	TGSCRR	0x18	2	TGS Counter Reload Register
CTU	CLCR1	0x1C	3	Commands List Cotrol Register 1
CTU	CLCR2	0x20	3	Commands List Cotrol Register 2
CTU	THCR1	0x24	3	Trigger Handler Control Register 1
CTU	THCR2	0x28	3	Trigger Handler Control Register 2
CTU	CLR_A_1	0x2C	2	Commad List Register 1
CTU	CLR_A_2	0x2E	2	Commad List Register 2
CTU	CLR_A_3	0x30	2	Commad List Register 3
CTU	CLR_A_4	0x32	2	Commad List Register 4
CTU	CLR_A_5	0x34	2	Commad List Register 5
CTU	CLR_A_6	0x36	2	Commad List Register 6
CTU	CLR_A_7	0x38	2	Commad List Register 7
CTU	CLR_A_8	0x3A	2	Commad List Register 8
CTU	CLR_A_9	0x3C	2	Commad List Register 9
CTU	CLR_A_10	0x3E	2	Commad List Register 10
CTU	CLR_A_11	0x40	2	Commad List Register 11
CTU	CLR_A_12	0x42	2	Commad List Register 12
CTU	CLR_A_13	0x44	2	Commad List Register 13
CTU	CLR_A_14	0x46	2	Commad List Register 14
CTU	CLR_A_15	0x48	2	Commad List Register 15
CTU	CLR_A_16	0x4A	2	Commad List Register 16
CTU	CLR_A_17	0x4C	2	Commad List Register 17
CTU	CLR_A_18	0x4E	2	Commad List Register 18
CTU	CLR_A_19	0x50	2	Commad List Register 19
CTU	CLR_A_20	0x52	2	Commad List Register 20
CTU	CLR_A_21	0x54	2	Commad List Register 21
CTU	CLR_A_22	0x56	2	Commad List Register 22
CTU	CLR_A_23	0x58	2	Commad List Register 23
CTU	CLR_A_24	0x5A	2	Commad List Register 24
CTU	FDCR	0x6C	2	Control register
CTU	FCR	0x70	3	Control register FIFO
CTU	FTH	0x74	3	Threshold1
CTU	IR	0xC4	2	Cross Triggering Unit Interrupt/DMA Register
CTU	COTR	0xC6	2	Control On-Time Register
CTU	CR	0xC8	2	Cross Triggering Unit Control Register
CTU	DFR	0xCA	2	Cross Triggering Unit digital Filter
CTU	EXPAR	0xCC	2	Cross Triggering Unit Expected value

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CTU	EXPBR	0xCE	2	Cross Triggering Unit Expected value
CTU	CNTRNGR	0xD0	2	Cross Triggering Unit conter range
CTU	LISTCSR	0xD4	3	List Mode register
LFAST	MCR	0x00	3	LFAST Mode Configuration Register
LFAST	SCR	0x04	3	LFAST Speed Control Register
LFAST	COCR	0x08	3	LFAST Correlator Control Register
LFAST	TMCR	0x0c	3	LFAST Test Mode Control Register
LFAST	ALCR	0x10	3	LFAST Auto Loopback Control Register
LFAST	RCDCR	0x14	3	LFAST Rate Change Delay Control Register
LFAST	SLCR	0x18	3	LFAST Wakeup Delay Control Register
LFAST	ICR	0x1c	3	LFAST ICLC Control Register
LFAST	PICR	0x20	3	LFAST Ping Control Register
LFAST	RFCR	0x2c	3	LFAST Rx FIFO CTS Control Register
LFAST	TIER	0x30	3	LFAST Tx Interrupt Enable Register
LFAST	RIER	0x34	3	LFAST Rx Interrupt Enable Register
LFAST	RIIER	0x38	3	LFAST Rx ICLC Interrupt Enable Register
LFAST	PLLCR	0x3c	3	LFAST PLL Control Register
DMAMUX_0	CHCFG0	0x00	1	configuration of DMA ch x
DMAMUX_0	CHCFG1	0x01	1	configuration of DMA ch x
DMAMUX_0	CHCFG2	0x02	1	configuration of DMA ch x
DMAMUX_0	CHCFG3	0x03	1	configuration of DMA ch x
DMAMUX_0	CHCFG4	0x04	1	configuration of DMA ch x
DMAMUX_0	CHCFG5	0x05	1	configuration of DMA ch x
DMAMUX_0	CHCFG6	0x06	1	configuration of DMA ch x
DMAMUX_0	CHCFG7	0x07	1	configuration of DMA ch x
DMAMUX_0	CHCFG8	0x08	1	configuration of DMA ch x
DMAMUX_0	CHCFG9	0x09	1	configuration of DMA ch x
DMAMUX_0	CHCFG10	0x0A	1	configuration of DMA ch x
DMAMUX_0	CHCFG11	0x0B	1	configuration of DMA ch x
DMAMUX_0	CHCFG12	0x0C	1	configuration of DMA ch x
DMAMUX_0	CHCFG13	0x0D	1	configuration of DMA ch x
DMAMUX_0	CHCFG14	0x0E	1	configuration of DMA ch x
DMAMUX_0	CHCFG15	0x0F	1	configuration of DMA ch x
DMAMUX_1	CHCFG0	0x00	1	configuration of DMA ch x
DMAMUX_1	CHCFG1	0x01	1	configuration of DMA ch x
DMAMUX_1	CHCFG2	0x02	1	configuration of DMA ch x
DMAMUX_1	CHCFG3	0x03	1	configuration of DMA ch x
DMAMUX_1	CHCFG4	0x04	1	configuration of DMA ch x
DMAMUX_1	CHCFG5	0x05	1	configuration of DMA ch x

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
DMAMUX_1	CHCFG6	0x06	1	configuration of DMA ch x
DMAMUX_1	CHCFG7	0x07	1	configuration of DMA ch x
DMAMUX_1	CHCFG8	0x08	1	configuration of DMA ch x
DMAMUX_1	CHCFG9	0x09	1	configuration of DMA ch x
DMAMUX_1	CHCFG10	0x0A	1	configuration of DMA ch x
DMAMUX_1	CHCFG11	0x0B	1	configuration of DMA ch x
DMAMUX_1	CHCFG12	0x0C	1	configuration of DMA ch x
DMAMUX_1	CHCFG13	0x0D	1	configuration of DMA ch x
DMAMUX_1	CHCFG14	0x0E	1	configuration of DMA ch x
DMAMUX_1	CHCFG15	0x0F	1	configuration of DMA ch x
SPI_1	MCR	0x00	3	Module Configuration Register
SPI_1	TCR	0x08	3	Transfer Count Register
SPI_1	CTAR0	0x0C	3	Clock and Transfer Attributes Register
SPI_1	CTAR1	0x10	3	Clock and Transfer Attributes Register
SPI_1	CTAR2	0x14	3	Clock and Transfer Attributes Register
SPI_1	CTAR3	0x18	3	Clock and Transfer Attributes Register
SPI_1	CTAR4	0x1C	3	Clock and Transfer Attributes Register
SPI_1	CTAR5	0x20	3	Clock and Transfer Attributes Register
SPI_1	CTAR6	0x24	3	Clock and Transfer Attributes Register
SPI_1	CTAR7	0x28	3	Clock and Transfer Attributes Register
SPI_1	RSER	0x30	3	DMA/Interrupt Request Select and Enable Register
SPI_2	MCR	0x00	3	Module Configuration Register
SPI_2	TCR	0x08	3	Transfer Count Register
SPI_2	CTAR0	0x0C	3	Clock and Transfer Attributes Register
SPI_2	CTAR1	0x10	3	Clock and Transfer Attributes Register
SPI_2	CTAR2	0x14	3	Clock and Transfer Attributes Register
SPI_2	CTAR3	0x18	3	Clock and Transfer Attributes Register
SPI_2	CTAR4	0x1C	3	Clock and Transfer Attributes Register
SPI_2	CTAR5	0x20	3	Clock and Transfer Attributes Register
SPI_2	CTAR6	0x24	3	Clock and Transfer Attributes Register
SPI_2	CTAR7	0x28	3	Clock and Transfer Attributes Register
SPI_2	RSER	0x30	3	DMA/Interrupt Request Select and Enable Register
ENET_0	EIMR	0x08	3	Interrupt Mask Register
ENET_0	MSCR	0x44	3	MII Speed Control Register
ENET_0	RCR	0x84	3	Receive Control Register
ENET_0	TCR	0xC4	3	Tramit Control Register
ENET_0	PALR	0xE4	3	Physical Address Lower Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ENET_0	PAUR	0xE8	3	Physical Address Upper Register
ENET_0	OPD	0xEC	3	Physical Address Upper Register
ENET_0	IAUR	0x118	3	Descriptor Individual Upper Address Register
ENET_0	IALR	0x11C	3	Descriptor Individual Lower Address Register
ENET_0	GAUR	0x120	3	Descriptor Group Upper Address Register
ENET_0	GALR	0x124	3	Descriptor Group Lower Address Register
ENET_0	TFWR	0x144	3	Transmit FIFO Watermark Register
ENET_0	RDSR	0x180	3	Receive Descriptor Ring Start Register
ENET_0	TDSR	0x184	3	Transmit Descriptor Ring Start Register
ENET_0	MRBR	0x188	3	Maximum Receive Buffer Size Register
ENET_0	RSFL	0x190	3	Receive FIFO Section Full Threshold
ENET_0	RSEM	0x194	3	Receive FIFO Section Empty Threshold
ENET_0	RAEM	0x198	3	Receive FIFO Almost Empty Threshold
ENET_0	RAFL	0x19C	3	Receive FIFO Almost Full Threshold
ENET_0	TSEM	0x1A0	3	Transmit FIFO Section Empty Threshold
ENET_0	TAEM	0x1A4	3	Transmit FIFO Almost Empty Threshold
ENET_0	TAFL	0x1A8	3	Transmit FIFO Almost Full Threshold
ENET_0	TIPG	0x1AC	3	Transmit Inter-Packet GAP
ENET_0	FTRL	0x1B0	3	Frame Truncate Length
ENET_0	TACC	0x1C0	3	Transmit Accelerator Function Configuration
ENET_0	RACC	0x1C4	3	Receive Accelerator Function Configuration
ENET_0	ATPER	0x40C	3	Timer Period Register
ENET_0	ATCOR	0x410	3	Timer Correction Register
ENET_0	ATINC	0x414	3	Time-Stamping Clock Period Register
ETIMER_1	CH0_COMP1	0x00	2	can be used to configure a pwm frequency
ETIMER_1	CH0_COMP2	0x02	2	can be used to configure a pwm frequency
ETIMER_1	CH0_LOAD	0x08	2	value to initialize the counter
ETIMER_1	CH0_CTRL1	0x0E	2	Channel Control Register
ETIMER_1	CH0_CTRL2	0x10	2	Channel Control Register
ETIMER_1	CH0_CTRL3	0x12	2	Channel Control Register
ETIMER_1	CH0_INTDMA	0x16	2	interrupt and dma enable
ETIMER_1	CH0_CMPLD1	0x18	2	can be used to configure a pwm frequency
ETIMER_1	CH0_CMPLD2	0x1A	2	can be used to configure a pwm frequency
ETIMER_1	CH0_CCCTRL	0x1C	2	Channel Compare and Capture Control Register
ETIMER_1	CH0_FILT	0x1E	2	input filter configuration
ETIMER_1	CH1_COMP1	0x20	2	can be used to configure a pwm frequency
ETIMER_1	CH1_COMP2	0x22	2	can be used to configure a pwm frequency
ETIMER_1	CH1_LOAD	0x28	2	value to initialize the counter

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ETIMER_1	CH1_CTRL1	0x2E	2	Channel Control Register
ETIMER_1	CH1_CTRL2	0x30	2	Channel Control Register
ETIMER_1	CH1_CTRL3	0x32	2	Channel Control Register
ETIMER_1	CH1_INTDMA	0x36	2	interrupt and dma enable
ETIMER_1	CH1_CMPLD1	0x38	2	can be used to configure a pwm frequency
ETIMER_1	CH1_CMPLD2	0x3A	2	can be used to configure a pwm frequency
ETIMER_1	CH1_CCCTRL	0x3C	2	Channel Compare and Capture Control Register
ETIMER_1	CH1_FILT	0x3E	2	input filter configuration
ETIMER_1	CH2_COMP1	0x40	2	can be used to configure a pwm frequency
ETIMER_1	CH2_COMP2	0x42	2	can be used to configure a pwm frequency
ETIMER_1	CH2_LOAD	0x48	2	value to initialize the counter
ETIMER_1	CH2_CTRL1	0x4E	2	Channel Control Register
ETIMER_1	CH2_CTRL2	0x50	2	Channel Control Register
ETIMER_1	CH2_CTRL3	0x52	2	Channel Control Register
ETIMER_1	CH2_INTDMA	0x56	2	interrupt and dma enable
ETIMER_1	CH2_CMPLD1	0x58	2	can be used to configure a pwm frequency
ETIMER_1	CH2_CMPLD2	0x5A	2	can be used to configure a pwm frequency
ETIMER_1	CH2_CCCTRL	0x5C	2	Channel Compare and Capture Control Register
ETIMER_1	CH2_FILT	0x5E	2	input filter configuration
ETIMER_1	CH3_COMP1	0x60	2	can be used to configure a pwm frequency
ETIMER_1	CH3_COMP2	0x62	2	can be used to configure a pwm frequency
ETIMER_1	CH3_LOAD	0x68	2	value to initialize the counter
ETIMER_1	CH3_CTRL1	0x6E	2	Channel Control Register
ETIMER_1	CH3_CTRL2	0x70	2	Channel Control Register
ETIMER_1	CH3_CTRL3	0x72	2	Channel Control Register
ETIMER_1	CH3_INTDMA	0x76	2	interrupt and dma enable
ETIMER_1	CH3_CMPLD1	0x78	2	can be used to configure a pwm frequency
ETIMER_1	CH3_CMPLD2	0x7A	2	can be used to configure a pwm frequency
ETIMER_1	CH3_CCCTRL	0x7C	2	Channel Compare and Capture Control Register
ETIMER_1	CH3_FILT	0x7E	2	input filter configuration
ETIMER_1	CH4_COMP1	0x80	2	can be used to configure a pwm frequency
ETIMER_1	CH4_COMP2	0x82	2	can be used to configure a pwm frequency
ETIMER_1	CH4_LOAD	0x88	2	value to initialize the counter
ETIMER_1	CH4_CTRL1	0x8E	2	Channel Control Register
ETIMER_1	CH4_CTRL2	0x90	2	Channel Control Register
ETIMER_1	CH4_CTRL3	0x92	2	Channel Control Register
ETIMER_1	CH4_INTDMA	0x96	2	interrupt and dma enable

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ETIMER_1	CH4_CMPLD1	0x98	2	can be used to configure a pwm frequency
ETIMER_1	CH4_CMPLD2	0x9A	2	can be used to configure a pwm frequency
ETIMER_1	CH4_CCCTRL	0x9C	2	Channel Compare and Capture Control Register
ETIMER_1	CH4_FILT	0x9E	2	input filter configuration
ETIMER_1	CH5_COMP1	0xA0	2	can be used to configure a pwm frequency
ETIMER_1	CH5_COMP2	0xA2	2	can be used to configure a pwm frequency
ETIMER_1	CH5_LOAD	0xA8	2	value to initialize the counter
ETIMER_1	CH5_CTRL1	0xAE	2	Channel Control Register
ETIMER_1	CH5_CTRL2	0xB0	2	Channel Control Register
ETIMER_1	CH5_CTRL3	0xB2	2	Channel Control Register
ETIMER_1	CH5_INTDMA	0xB6	2	interrupt and dma enable
ETIMER_1	CH5_CMPLD1	0xB8	2	can be used to configure a pwm frequency
ETIMER_1	CH5_CMPLD2	0xBA	2	can be used to configure a pwm frequency
ETIMER_1	CH5_CCCTRL	0xBC	2	Channel Compare and Capture Control Register
ETIMER_1	CH5_FILT	0xBE	2	input filter configuration
ETIMER_1	CH6_COMP1	0xC0	2	can be used to configure a pwm frequency
ETIMER_1	CH6_COMP2	0xC2	2	can be used to configure a pwm frequency
ETIMER_1	CH6_LOAD	0xC8	2	value to initialize the counter
ETIMER_1	CH6_CTRL1	0xCE	2	Channel Control Register
ETIMER_1	CH6_CTRL2	0xD0	2	Channel Control Register
ETIMER_1	CH6_CTRL3	0xD2	2	Channel Control Register
ETIMER_1	CH6_INTDMA	0xD6	2	interrupt and dma enable
ETIMER_1	CH6_CMPLD1	0xD8	2	can be used to configure a pwm frequency
ETIMER_1	CH6_CMPLD2	0xDA	2	can be used to configure a pwm frequency
ETIMER_1	CH6_CCCTRL	0xDC	2	Channel Compare and Capture Control Register
ETIMER_1	CH6_FILT	0xDE	2	input filter configuration
ETIMER_1	ENBL	0x10C	2	channel enable register
ETIMER_1	DREQ0	0x110	2	dma req_0 configuration
ETIMER_1	DREQ1	0x112	2	dma req_1 configuration
ETIMER_2	CH0_COMP1	0x00	2	can be used to configure a pwm frequency
ETIMER_2	CH0_COMP2	0x02	2	can be used to configure a pwm frequency
ETIMER_2	CH0_LOAD	0x08	2	value to initialize the counter
ETIMER_2	CH0_CTRL1	0x0E	2	Channel Control Register
ETIMER_2	CH0_CTRL2	0x10	2	Channel Control Register
ETIMER_2	CH0_CTRL3	0x12	2	Channel Control Register
ETIMER_2	CH0_INTDMA	0x16	2	interrupt and dma enable
ETIMER_2	CH0_CMPLD1	0x18	2	can be used to configure a pwm frequency

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ETIMER_2	CH0_CMPLD2	0x1A	2	can be used to configure a pwm frequency
ETIMER_2	CH0_CCCTRL	0x1C	2	Channel Compare and Capture Control Register
ETIMER_2	CH0_FILT	0x1E	2	input filter configuration
ETIMER_2	CH1_COMP1	0x20	2	can be used to configure a pwm frequency
ETIMER_2	CH1_COMP2	0x22	2	can be used to configure a pwm frequency
ETIMER_2	CH1_LOAD	0x28	2	value to initialize the counter
ETIMER_2	CH1_CTRL1	0x2E	2	Channel Control Register
ETIMER_2	CH1_CTRL2	0x30	2	Channel Control Register
ETIMER_2	CH1_CTRL3	0x32	2	Channel Control Register
ETIMER_2	CH1_INTDMA	0x36	2	interrupt and dma enable
ETIMER_2	CH1_CMPLD1	0x38	2	can be used to configure a pwm frequency
ETIMER_2	CH1_CMPLD2	0x3A	2	can be used to configure a pwm frequency
ETIMER_2	CH1_CCCTRL	0x3C	2	Channel Compare and Capture Control Register
ETIMER_2	CH1_FILT	0x3E	2	input filter configuration
ETIMER_2	CH2_COMP1	0x40	2	can be used to configure a pwm frequency
ETIMER_2	CH2_COMP2	0x42	2	can be used to configure a pwm frequency
ETIMER_2	CH2_LOAD	0x48	2	value to initialize the counter
ETIMER_2	CH2_CTRL1	0x4E	2	Channel Control Register
ETIMER_2	CH2_CTRL2	0x50	2	Channel Control Register
ETIMER_2	CH2_CTRL3	0x52	2	Channel Control Register
ETIMER_2	CH2_INTDMA	0x56	2	interrupt and dma enable
ETIMER_2	CH2_CMPLD1	0x58	2	can be used to configure a pwm frequency
ETIMER_2	CH2_CMPLD2	0x5A	2	can be used to configure a pwm frequency
ETIMER_2	CH2_CCCTRL	0x5C	2	Channel Compare and Capture Control Register
ETIMER_2	CH2_FILT	0x5E	2	input filter configuration
ETIMER_2	CH3_COMP1	0x60	2	can be used to configure a pwm frequency
ETIMER_2	CH3_COMP2	0x62	2	can be used to configure a pwm frequency
ETIMER_2	CH3_LOAD	0x68	2	value to initialize the counter
ETIMER_2	CH3_CTRL1	0x6E	2	Channel Control Register
ETIMER_2	CH3_CTRL2	0x70	2	Channel Control Register
ETIMER_2	CH3_CTRL3	0x72	2	Channel Control Register
ETIMER_2	CH3_INTDMA	0x76	2	interrupt and dma enable
ETIMER_2	CH3_CMPLD1	0x78	2	can be used to configure a pwm frequency
ETIMER_2	CH3_CMPLD2	0x7A	2	can be used to configure a pwm frequency
ETIMER_2	CH3_CCCTRL	0x7C	2	Channel Compare and Capture Control Register
ETIMER_2	CH3_FILT	0x7E	2	input filter configuration

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
ETIMER_2	CH4_COMP1	0x80	2	can be used to configure a pwm frequency
ETIMER_2	CH4_COMP2	0x82	2	can be used to configure a pwm frequency
ETIMER_2	CH4_LOAD	0x88	2	value to initialize the counter
ETIMER_2	CH4_CTRL1	0x8E	2	Channel Control Register
ETIMER_2	CH4_CTRL2	0x90	2	Channel Control Register
ETIMER_2	CH4_CTRL3	0x92	2	Channel Control Register
ETIMER_2	CH4_INTDMA	0x96	2	interrupt and dma enable
ETIMER_2	CH4_CMPLD1	0x98	2	can be used to configure a pwm frequency
ETIMER_2	CH4_CMPLD2	0x9A	2	can be used to configure a pwm frequency
ETIMER_2	CH4_CCCTRL	0x9C	2	Channel Compare and Capture Control Register
ETIMER_2	CH4_FILT	0x9E	2	input filter configuration
ETIMER_2	CH5_COMP1	0xA0	2	can be used to configure a pwm frequency
ETIMER_2	CH5_COMP2	0xA2	2	can be used to configure a pwm frequency
ETIMER_2	CH5_LOAD	0xA8	2	value to initialize the counter
ETIMER_2	CH5_CTRL1	0xAE	2	Channel Control Register
ETIMER_2	CH5_CTRL2	0xB0	2	Channel Control Register
ETIMER_2	CH5_CTRL3	0xB2	2	Channel Control Register
ETIMER_2	CH5_INTDMA	0xB6	2	interrupt and dma enable
ETIMER_2	CH5_CMPLD1	0xB8	2	can be used to configure a pwm frequency
ETIMER_2	CH5_CMPLD2	0xBA	2	can be used to configure a pwm frequency
ETIMER_2	CH5_CCCTRL	0xBC	2	Channel Compare and Capture Control Register
ETIMER_2	CH5_FILT	0xBE	2	input filter configuration
ETIMER_2	CH6_COMP1	0xC0	2	can be used to configure a pwm frequency
ETIMER_2	CH6_COMP2	0xC2	2	can be used to configure a pwm frequency
ETIMER_2	CH6_LOAD	0xC8	2	value to initialize the counter
ETIMER_2	CH6_CTRL1	0xCE	2	Channel Control Register
ETIMER_2	CH6_CTRL3	0xD2	2	Channel Control Register
ETIMER_2	CH6_INTDMA	0xD6	2	interrupt and dma enable
ETIMER_2	CH6_CMPLD1	0xD8	2	can be used to configure a pwm frequency
ETIMER_2	CH6_CMPLD2	0xDA	2	can be used to configure a pwm frequency
ETIMER_2	CH6_CCCTRL	0xDC	2	Channel Compare and Capture Control Register
ETIMER_2	CH6_FILT	0xDE	2	input filter configuration
ETIMER_2	ENBL	0x10C	2	channel enable register
ETIMER_2	DREQ0	0x110	2	dma req_0 configuration
ETIMER_2	DREQ1	0x112	2	dma req_1 configuration
FCCU	CFG	0x08	3	global configuration of the macrocell (writable only in CONFIG state)

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FCCU	NCF_CFG0	0x1C	3	not critical fault configuration (writable only in CONFIG state)
FCCU	NCF_CFG1	0x20	3	not critical fault configuration (writable only in CONFIG state)
FCCU	NCF_CFG2	0x24	3	not critical fault configuration (writable only in CONFIG state)
FCCU	NCF_CFG3	0x28	3	not critical fault configuration (writable only in CONFIG state)
FCCU	NCFS_CFG0	0x4C	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG1	0x50	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG2	0x54	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG3	0x58	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG4	0x5C	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG5	0x60	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG6	0x64	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCFS_CFG7	0x68	3	Not Critical fault state configuration (writable only in CONFIG state)
FCCU	NCF_E0	0x94	3	Non-critical fault enable (writable only in CONFIG state)
FCCU	NCF_E1	0x98	3	Non-critical fault enable (writable only in CONFIG state)
FCCU	NCF_E2	0x9C	3	Non-critical fault enable (writable only in CONFIG state)
FCCU	NCF_E3	0xA0	3	Non-critical fault enable (writable only in CONFIG state)
FCCU	NCF_TOE0	0xA4	3	Non-critical fault time-out enable (writable only in CONFIG state)
FCCU	NCF_TOE1	0xA8	3	Non-critical fault time-out enable (writable only in CONFIG state)
FCCU	NCF_TOE2	0xAC	3	Non-critical fault time-out enable (writable only in CONFIG state)
FCCU	NCF_TOE3	0xB0	3	Non-critical fault time-out enable (writable only in CONFIG state)
FCCU	NCF_TO	0xB4	3	Non-critical fault time-out (writable only in CONFIG state)
FCCU	CFG_TO	0xB8	3	configuration time out
FCCU	NCFF	0xDC	3	injection of critical fault (protected to avoid that software can inject a fault)

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FCCU	IRQ_EN	0xE4	3	irq enable register
C55FMC	MCR	0x00	3	Module Configuration Register
C55FMC	LOCK0	0x10	3	Lock 0 register
C55FMC	LOCK1	0x14	3	Lock 1 register
C55FMC	LOCK2	0x18	3	Lock 2 register
C55FMC	UT0	0x54	3	UTest 0 register
C55FMC	UM0	0x58	3	UMISR register
C55FMC	UM1	0x5C	3	UMISR register
C55FMC	UM2	0x60	3	UMISR register
C55FMC	UM3	0x64	3	UMISR register
C55FMC	UM4	0x68	3	UMISR register
C55FMC	UM5	0x6C	3	UMISR register
C55FMC	UM6	0x70	3	UMISR register
C55FMC	UM7	0x74	3	UMISR register
C55FMC	UM8	0x78	3	UMISR register
C55FMC	UM9	0x7C	3	UMISR register
CAN_0	MCR	0x0	3	Module Configuration
CAN_0	CTRL1	0x4	3	Control Register
CAN_0	RXMGMASK	0x10	3	Rx Global Mask
CAN_0	RX14MASK	0x14	3	Rx Buffer 14 Mask
CAN_0	RX15MASK	0x18	3	Rx Buffer 15 Mask
CAN_0	IMASK2	0x24	3	Interrupt Masks 2
CAN_0	IMASK1	0x28	3	Interrupt Masks 1
CAN_0	CTRL2	0x34	3	Control 2
CAN_0	CS0	0x80	3	CS[31-24] should be protected
CAN_0	ID0	0x84	3	ID Register of Message Buffer 0
CAN_0	CS1	0x90	3	CS[31-24] should be protected
CAN_0	ID1	0x94	3	ID Register of Message Buffer 1
CAN_0	CS2	0xA0	3	CS[31-24] should be protected
CAN_0	ID2	0xA4	3	ID Register of Message Buffer 2
CAN_0	CS3	0xB0	3	CS[31-24] should be protected
CAN_0	ID3	0xB4	3	ID Register of Message Buffer 3
CAN_0	CS4	0xC0	3	CS[31-24] should be protected
CAN_0	ID4	0xC4	3	ID Register of Message Buffer 4
CAN_0	CS5	0xD0	3	CS[31-24] should be protected
CAN_0	ID5	0xD4	3	ID Register of Message Buffer 5
CAN_0	CS6	0xE0	3	CS[31-24] should be protected
CAN_0	ID6	0xE4	3	ID Register of Message Buffer 6
CAN_0	CS7	0xF0	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	ID7	0xF4	3	ID Register of Message Buffer 7
CAN_0	CS8	0x100	3	CS[31-24] should be protected
CAN_0	ID8	0x104	3	ID Register of Message Buffer 8
CAN_0	CS9	0x110	3	CS[31-24] should be protected
CAN_0	ID9	0x114	3	ID Register of Message Buffer 9
CAN_0	CS10	0x120	3	CS[31-24] should be protected
CAN_0	ID10	0x124	3	ID Register of Message Buffer 10
CAN_0	CS11	0x130	3	CS[31-24] should be protected
CAN_0	ID11	0x134	3	ID Register of Message Buffer 11
CAN_0	CS12	0x140	3	CS[31-24] should be protected
CAN_0	ID12	0x144	3	ID Register of Message Buffer 12
CAN_0	CS13	0x150	3	CS[31-24] should be protected
CAN_0	ID13	0x154	3	ID Register of Message Buffer 13
CAN_0	CS14	0x160	3	CS[31-24] should be protected
CAN_0	ID14	0x164	3	ID Register of Message Buffer 14
CAN_0	CS15	0x170	3	CS[31-24] should be protected
CAN_0	ID15	0x174	3	ID Register of Message Buffer 15
CAN_0	CS16	0x180	3	CS[31-24] should be protected
CAN_0	ID16	0x184	3	ID Register of Message Buffer 16
CAN_0	CS17	0x190	3	CS[31-24] should be protected
CAN_0	ID17	0x194	3	ID Register of Message Buffer 17
CAN_0	CS18	0x1A0	3	CS[31-24] should be protected
CAN_0	ID18	0x1A4	3	ID Register of Message Buffer 18
CAN_0	CS19	0x1B0	3	CS[31-24] should be protected
CAN_0	ID19	0x1B4	3	ID Register of Message Buffer 19
CAN_0	CS20	0x1C0	3	CS[31-24] should be protected
CAN_0	ID20	0x1C4	3	ID Register of Message Buffer 20
CAN_0	CS21	0x1D0	3	CS[31-24] should be protected
CAN_0	ID21	0x1D4	3	ID Register of Message Buffer 21
CAN_0	CS22	0x1E0	3	CS[31-24] should be protected
CAN_0	ID22	0x1E4	3	ID Register of Message Buffer 22
CAN_0	CS23	0x1F0	3	CS[31-24] should be protected
CAN_0	ID23	0x1F4	3	ID Register of Message Buffer 23
CAN_0	CS24	0x200	3	CS[31-24] should be protected
CAN_0	ID24	0x204	3	ID Register of Message Buffer 24
CAN_0	CS25	0x210	3	CS[31-24] should be protected
CAN_0	ID25	0x214	3	ID Register of Message Buffer 25
CAN_0	CS26	0x220	3	CS[31-24] should be protected
CAN_0	ID26	0x224	3	ID Register of Message Buffer 26

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	CS27	0x230	3	CS[31-24] should be protected
CAN_0	ID27	0x234	3	ID Register of Message Buffer 27
CAN_0	CS28	0x240	3	CS[31-24] should be protected
CAN_0	ID28	0x244	3	ID Register of Message Buffer 28
CAN_0	CS29	0x250	3	CS[31-24] should be protected
CAN_0	ID29	0x254	3	ID Register of Message Buffer 29
CAN_0	CS30	0x260	3	CS[31-24] should be protected
CAN_0	ID30	0x264	3	ID Register of Message Buffer 30
CAN_0	CS31	0x270	3	CS[31-24] should be protected
CAN_0	ID31	0x274	3	ID Register of Message Buffer 31
CAN_0	CS32	0x280	3	CS[31-24] should be protected
CAN_0	ID32	0x284	3	ID Register of Message Buffer 32
CAN_0	CS33	0x290	3	CS[31-24] should be protected
CAN_0	ID33	0x294	3	ID Register of Message Buffer 33
CAN_0	CS34	0x2a0	3	CS[31-24] should be protected
CAN_0	ID34	0x2a4	3	ID Register of Message Buffer 34
CAN_0	CS35	0x2b0	3	CS[31-24] should be protected
CAN_0	ID35	0x2b4	3	ID Register of Message Buffer 35
CAN_0	CS36	0x2c0	3	CS[31-24] should be protected
CAN_0	ID36	0x2c4	3	ID Register of Message Buffer 36
CAN_0	CS37	0x2d0	3	CS[31-24] should be protected
CAN_0	ID37	0x2d4	3	ID Register of Message Buffer 37
CAN_0	CS38	0x2E0	3	CS[31-24] should be protected
CAN_0	ID38	0x2E4	3	ID Register of Message Buffer 38
CAN_0	CS39	0x2f0	3	CS[31-24] should be protected
CAN_0	ID39	0x2f4	3	ID Register of Message Buffer 39
CAN_0	CS40	0x300	3	CS[31-24] should be protected
CAN_0	ID40	0x304	3	ID Register of Message Buffer 40
CAN_0	CS41	0x310	3	CS[31-24] should be protected
CAN_0	ID41	0x314	3	ID Register of Message Buffer 41
CAN_0	CS42	0x320	3	CS[31-24] should be protected
CAN_0	ID42	0x324	3	ID Register of Message Buffer 42
CAN_0	CS43	0x330	3	CS[31-24] should be protected
CAN_0	ID43	0x334	3	ID Register of Message Buffer 43
CAN_0	CS44	0x340	3	CS[31-24] should be protected
CAN_0	ID44	0x344	3	ID Register of Message Buffer 44
CAN_0	CS45	0x350	3	CS[31-24] should be protected
CAN_0	ID45	0x354	3	ID Register of Message Buffer 45
CAN_0	CS46	0x360	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	ID46	0x364	3	ID Register of Message Buffer 46
CAN_0	CS47	0x370	3	CS[31-24] should be protected
CAN_0	ID47	0x374	3	ID Register of Message Buffer 47
CAN_0	CS48	0x380	3	CS[31-24] should be protected
CAN_0	ID48	0x384	3	ID Register of Message Buffer 48
CAN_0	CS49	0x390	3	CS[31-24] should be protected
CAN_0	ID49	0x394	3	ID Register of Message Buffer 49
CAN_0	CS50	0x3a0	3	CS[31-24] should be protected
CAN_0	ID50	0x3a4	3	ID Register of Message Buffer 50
CAN_0	CS51	0x3b0	3	CS[31-24] should be protected
CAN_0	ID51	0x3b4	3	ID Register of Message Buffer 51
CAN_0	CS52	0x3c0	3	CS[31-24] should be protected
CAN_0	ID52	0x3c4	3	ID Register of Message Buffer 52
CAN_0	CS53	0x3d0	3	CS[31-24] should be protected
CAN_0	ID53	0x3d4	3	ID Register of Message Buffer 53
CAN_0	CS54	0x3E0	3	CS[31-24] should be protected
CAN_0	ID54	0x3E4	3	ID Register of Message Buffer 54
CAN_0	CS55	0x3f0	3	CS[31-24] should be protected
CAN_0	ID55	0x3f4	3	ID Register of Message Buffer 55
CAN_0	CS56	0x400	3	CS[31-24] should be protected
CAN_0	ID56	0x404	3	ID Register of Message Buffer 56
CAN_0	CS57	0x410	3	CS[31-24] should be protected
CAN_0	ID57	0x414	3	ID Register of Message Buffer 57
CAN_0	CS58	0x420	3	CS[31-24] should be protected
CAN_0	ID58	0x424	3	ID Register of Message Buffer 58
CAN_0	CS59	0x430	3	CS[31-24] should be protected
CAN_0	ID59	0x434	3	ID Register of Message Buffer 59
CAN_0	CS60	0x440	3	CS[31-24] should be protected
CAN_0	ID60	0x444	3	ID Register of Message Buffer 60
CAN_0	CS61	0x450	3	CS[31-24] should be protected
CAN_0	ID61	0x454	3	ID Register of Message Buffer 61
CAN_0	CS62	0x460	3	CS[31-24] should be protected
CAN_0	ID62	0x464	3	ID Register of Message Buffer 62
CAN_0	CS63	0x470	3	CS[31-24] should be protected
CAN_0	ID63	0x474	3	ID Register of Message Buffer 63
CAN_0	CS64	0x480	3	CS[31-24] should be protected
CAN_0	ID64	0x484	3	ID Register of Message Buffer 64
CAN_0	CS65	0x490	3	CS[31-24] should be protected
CAN_0	ID65	0x494	3	ID Register of Message Buffer 65

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	CS66	0x4A0	3	CS[31-24] should be protected
CAN_0	ID66	0x4A4	3	ID Register of Message Buffer 66
CAN_0	CS67	0x4B0	3	CS[31-24] should be protected
CAN_0	ID67	0x4B4	3	ID Register of Message Buffer 67
CAN_0	ID68	0x4C4	3	ID Register of Message Buffer 68
CAN_0	CS69	0x4D0	3	CS[31-24] should be protected
CAN_0	ID69	0x4D4	3	ID Register of Message Buffer 69
CAN_0	ID70	0x4E4	3	ID Register of Message Buffer 70
CAN_0	CS71	0x4F0	3	CS[31-24] should be protected
CAN_0	ID71	0x4F4	3	ID Register of Message Buffer 71
CAN_0	CS72	0x500	3	CS[31-24] should be protected
CAN_0	ID72	0x504	3	ID Register of Message Buffer 72
CAN_0	CS73	0x510	3	CS[31-24] should be protected
CAN_0	ID73	0x514	3	ID Register of Message Buffer 73
CAN_0	CS74	0x520	3	CS[31-24] should be protected
CAN_0	ID74	0x524	3	ID Register of Message Buffer 74
CAN_0	CS75	0x530	3	CS[31-24] should be protected
CAN_0	ID75	0x534	3	ID Register of Message Buffer 75
CAN_0	CS76	0x540	3	CS[31-24] should be protected
CAN_0	ID76	0x544	3	ID Register of Message Buffer 76
CAN_0	CS77	0x550	3	CS[31-24] should be protected
CAN_0	ID77	0x554	3	ID Register of Message Buffer 77
CAN_0	CS78	0x560	3	CS[31-24] should be protected
CAN_0	ID78	0x564	3	ID Register of Message Buffer 78
CAN_0	CS79	0x570	3	CS[31-24] should be protected
CAN_0	ID79	0x574	3	ID Register of Message Buffer 79
CAN_0	CS80	0x580	3	CS[31-24] should be protected
CAN_0	ID80	0x584	3	ID Register of Message Buffer 80
CAN_0	CS81	0x590	3	CS[31-24] should be protected
CAN_0	ID81	0x594	3	ID Register of Message Buffer 81
CAN_0	CS82	0x5A0	3	CS[31-24] should be protected
CAN_0	ID82	0x5A4	3	ID Register of Message Buffer 82
CAN_0	CS83	0x5B0	3	CS[31-24] should be protected
CAN_0	ID83	0x5B4	3	ID Register of Message Buffer 83
CAN_0	CS84	0x5C0	3	CS[31-24] should be protected
CAN_0	ID84	0x5C4	3	ID Register of Message Buffer 84
CAN_0	CS85	0x5D0	3	CS[31-24] should be protected
CAN_0	ID85	0x5D4	3	ID Register of Message Buffer 85
CAN_0	CS86	0x5E0	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	ID86	0x5E4	3	ID Register of Message Buffer 86
CAN_0	CS88	0x600	3	CS[31-24] should be protected
CAN_0	ID88	0x604	3	ID Register of Message Buffer 87
CAN_0	RXIMR0	0x880	3	Rx Individual Mask Register 0
CAN_0	RXIMR1	0x884	3	Rx Individual Mask Register 1
CAN_0	RXIMR2	0x888	3	Rx Individual Mask Register 2
CAN_0	RXIMR3	0x88C	3	Rx Individual Mask Register 3
CAN_0	RXIMR4	0x890	3	Rx Individual Mask Register 4
CAN_0	RXIMR5	0x894	3	Rx Individual Mask Register 5
CAN_0	RXIMR6	0x898	3	Rx Individual Mask Register 6
CAN_0	RXIMR7	0x89C	3	Rx Individual Mask Register 7
CAN_0	RXIMR8	0x8A0	3	Rx Individual Mask Register 8
CAN_0	RXIMR9	0x8A4	3	Rx Individual Mask Register 9
CAN_0	RXIMR10	0x8A8	3	Rx Individual Mask Register 10
CAN_0	RXIMR11	0x8AC	3	Rx Individual Mask Register 11
CAN_0	RXIMR12	0x8B0	3	Rx Individual Mask Register 12
CAN_0	RXIMR13	0x8B4	3	Rx Individual Mask Register 13
CAN_0	RXIMR14	0x8B8	3	Rx Individual Mask Register 14
CAN_0	RXIMR15	0x8BC	3	Rx Individual Mask Register 15
CAN_0	RXIMR16	0x8C0	3	Rx Individual Mask Register 16
CAN_0	RXIMR17	0x8C4	3	Rx Individual Mask Register 17
CAN_0	RXIMR18	0x8C8	3	Rx Individual Mask Register 18
CAN_0	RXIMR19	0x8CC	3	Rx Individual Mask Register 19
CAN_0	RXIMR20	0x8D0	3	Rx Individual Mask Register 20
CAN_0	RXIMR21	0x8D4	3	Rx Individual Mask Register 21
CAN_0	RXIMR22	0x8D8	3	Rx Individual Mask Register 22
CAN_0	RXIMR23	0x8DC	3	Rx Individual Mask Register 23
CAN_0	RXIMR24	0x8E0	3	Rx Individual Mask Register 24
CAN_0	RXIMR25	0x8E4	3	Rx Individual Mask Register 25
CAN_0	RXIMR26	0x8E8	3	Rx Individual Mask Register 26
CAN_0	RXIMR27	0x8EC	3	Rx Individual Mask Register 27
CAN_0	RXIMR28	0x8F0	3	Rx Individual Mask Register 28
CAN_0	RXIMR29	0x8F4	3	Rx Individual Mask Register 29
CAN_0	RXIMR30	0x8F8	3	Rx Individual Mask Register 30
CAN_0	RXIMR31	0x8FC	3	Rx Individual Mask Register 31
CAN_0	RXIMR32	0x900	3	Rx Individual Mask Register 32
CAN_0	RXIMR33	0x904	3	Rx Individual Mask Register 33
CAN_0	RXIMR34	0x908	3	Rx Individual Mask Register 34
CAN_0	RXIMR35	0x90C	3	Rx Individual Mask Register 35

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	RXIMR36	0x910	3	Rx Individual Mask Register 36
CAN_0	RXIMR37	0x914	3	Rx Individual Mask Register 37
CAN_0	RXIMR38	0x918	3	Rx Individual Mask Register 38
CAN_0	RXIMR39	0x91C	3	Rx Individual Mask Register 39
CAN_0	RXIMR40	0x920	3	Rx Individual Mask Register 40
CAN_0	RXIMR41	0x924	3	Rx Individual Mask Register 41
CAN_0	RXIMR42	0x928	3	Rx Individual Mask Register 42
CAN_0	RXIMR43	0x92C	3	Rx Individual Mask Register 43
CAN_0	RXIMR44	0x930	3	Rx Individual Mask Register 44
CAN_0	RXIMR45	0x934	3	Rx Individual Mask Register 45
CAN_0	RXIMR46	0x938	3	Rx Individual Mask Register 46
CAN_0	RXIMR47	0x93C	3	Rx Individual Mask Register 47
CAN_0	RXIMR48	0x940	3	Rx Individual Mask Register 48
CAN_0	RXIMR49	0x944	3	Rx Individual Mask Register 49
CAN_0	RXIMR50	0x948	3	Rx Individual Mask Register 50
CAN_0	RXIMR51	0x94C	3	Rx Individual Mask Register 51
CAN_0	RXIMR52	0x950	3	Rx Individual Mask Register 52
CAN_0	RXIMR53	0x954	3	Rx Individual Mask Register 53
CAN_0	RXIMR54	0x958	3	Rx Individual Mask Register 54
CAN_0	RXIMR55	0x95C	3	Rx Individual Mask Register 55
CAN_0	RXIMR56	0x960	3	Rx Individual Mask Register 56
CAN_0	RXIMR57	0x964	3	Rx Individual Mask Register 57
CAN_0	RXIMR58	0x968	3	Rx Individual Mask Register 58
CAN_0	RXIMR59	0x96C	3	Rx Individual Mask Register 59
CAN_0	RXIMR60	0x970	3	Rx Individual Mask Register 60
CAN_0	RXIMR61	0x974	3	Rx Individual Mask Register 61
CAN_0	RXIMR62	0x978	3	Rx Individual Mask Register 62
CAN_0	RXIMR63	0x97C	3	Rx Individual Mask Register 63
CAN_0	RXIMR64	0x980	3	Rx Individual Mask Register 64
CAN_0	RXIMR65	0x984	3	Rx Individual Mask Register 65
CAN_0	RXIMR66	0x988	3	Rx Individual Mask Register 66
CAN_0	RXIMR67	0x98C	3	Rx Individual Mask Register 67
CAN_0	RXIMR68	0x990	3	Rx Individual Mask Register 68
CAN_0	RXIMR69	0x994	3	Rx Individual Mask Register 69
CAN_0	RXIMR70	0x998	3	Rx Individual Mask Register 70
CAN_0	RXIMR71	0x99C	3	Rx Individual Mask Register 71
CAN_0	RXIMR72	0x9A0	3	Rx Individual Mask Register 72
CAN_0	RXIMR73	0x9A4	3	Rx Individual Mask Register 73
CAN_0	RXIMR74	0x9A8	3	Rx Individual Mask Register 74

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_0	RXIMR75	0x9AC	3	Rx Individual Mask Register 75
CAN_0	RXIMR76	0x9B0	3	Rx Individual Mask Register 76
CAN_0	RXIMR77	0x9B4	3	Rx Individual Mask Register 77
CAN_0	RXIMR78	0x9B8	3	Rx Individual Mask Register 78
CAN_0	RXIMR79	0x9BC	3	Rx Individual Mask Register 79
CAN_0	RXIMR80	0x9C0	3	Rx Individual Mask Register 80
CAN_0	RXIMR81	0x9C4	3	Rx Individual Mask Register 81
CAN_0	RXIMR82	0x9C8	3	Rx Individual Mask Register 82
CAN_0	RXIMR83	0x9CC	3	Rx Individual Mask Register 83
CAN_0	RXIMR84	0x9D0	3	Rx Individual Mask Register 84
CAN_0	RXIMR85	0x9D4	3	Rx Individual Mask Register 85
CAN_0	RXIMR86	0x9D8	3	Rx Individual Mask Register 86
CAN_0	RXIMR87	0x9DC	3	Rx Individual Mask Register 87
CAN_0	RXIMR88	0x9E0	3	Rx Individual Mask Register 88
CAN_0	RXIMR89	0x9E4	3	Rx Individual Mask Register 89
CAN_0	RXIMR90	0x9E8	3	Rx Individual Mask Register 90
CAN_0	RXIMR91	0x9EC	3	Rx Individual Mask Register 91
CAN_0	RXIMR92	0x9F0	3	Rx Individual Mask Register 92
CAN_0	RXIMR93	0x9F4	3	Rx Individual Mask Register 93
CAN_0	RXIMR94	0x9F8	3	Rx Individual Mask Register 94
CAN_0	RXIMR95	0x9FC	3	Rx Individual Mask Register 95
CAN_0	MECR	0xAE0	3	Memory error configuration register
CAN_0	FDCTRL	0xC00	3	CAN FD Control Register
CAN_0	FDCBT	0xC04	3	CAN FD Bit Timing Register
CAN_1	MCR	0x00	3	Module Configuration
CAN_1	CTRL1	0x04	3	Control Register
CAN_1	RXMGMASK	0x10	3	Rx Global Mask
CAN_1	RX14MASK	0x14	3	Rx Buffer 14 Mask
CAN_1	RX15MASK	0x18	3	Rx Buffer 15 Mask
CAN_1	IMASK2	0x24	3	Interrupt Masks 2
CAN_1	IMASK1	0x28	3	Interrupt Masks 1
CAN_1	CTRL2	0x34	3	Control 2
CAN_1	CS0	0x80	3	CS[31-24] should be protected
CAN_1	ID0	0x84	3	ID Register of Message Buffer 0
CAN_1	CS1	0x90	3	CS[31-24] should be protected
CAN_1	ID1	0x94	3	ID Register of Message Buffer 1
CAN_1	CS2	0xA0	3	CS[31-24] should be protected
CAN_1	ID2	0xA4	3	ID Register of Message Buffer 2
CAN_1	CS3	0xB0	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_1	ID3	0xB4	3	ID Register of Message Buffer 3
CAN_1	CS4	0xC0	3	CS[31-24] should be protected
CAN_1	ID4	0xC4	3	ID Register of Message Buffer 4
CAN_1	CS5	0xD0	3	CS[31-24] should be protected
CAN_1	ID5	0xD4	3	ID Register of Message Buffer 5
CAN_1	CS6	0xE0	3	CS[31-24] should be protected
CAN_1	ID6	0xE4	3	ID Register of Message Buffer 6
CAN_1	CS7	0xF0	3	CS[31-24] should be protected
CAN_1	ID7	0xF4	3	ID Register of Message Buffer 7
CAN_1	CS8	0x100	3	CS[31-24] should be protected
CAN_1	ID8	0x104	3	ID Register of Message Buffer 8
CAN_1	CS9	0x110	3	CS[31-24] should be protected
CAN_1	ID9	0x114	3	ID Register of Message Buffer 9
CAN_1	CS10	0x120	3	CS[31-24] should be protected
CAN_1	ID10	0x124	3	ID Register of Message Buffer 10
CAN_1	CS11	0x130	3	CS[31-24] should be protected
CAN_1	ID11	0x134	3	ID Register of Message Buffer 11
CAN_1	CS12	0x140	3	CS[31-24] should be protected
CAN_1	ID12	0x144	3	ID Register of Message Buffer 12
CAN_1	CS13	0x150	3	CS[31-24] should be protected
CAN_1	ID13	0x154	3	ID Register of Message Buffer 13
CAN_1	CS14	0x160	3	CS[31-24] should be protected
CAN_1	ID14	0x164	3	ID Register of Message Buffer 14
CAN_1	CS15	0x170	3	CS[31-24] should be protected
CAN_1	ID15	0x174	3	ID Register of Message Buffer 15
CAN_1	CS16	0x180	3	CS[31-24] should be protected
CAN_1	ID16	0x184	3	ID Register of Message Buffer 16
CAN_1	CS17	0x190	3	CS[31-24] should be protected
CAN_1	ID17	0x194	3	ID Register of Message Buffer 17
CAN_1	CS18	0x1A0	3	CS[31-24] should be protected
CAN_1	ID18	0x1A4	3	ID Register of Message Buffer 18
CAN_1	CS19	0x1B0	3	CS[31-24] should be protected
CAN_1	ID19	0x1B4	3	ID Register of Message Buffer 19
CAN_1	CS20	0x1C0	3	CS[31-24] should be protected
CAN_1	ID20	0x1C4	3	ID Register of Message Buffer 20
CAN_1	CS21	0x1D0	3	CS[31-24] should be protected
CAN_1	ID21	0x1D4	3	ID Register of Message Buffer 21
CAN_1	CS22	0x1E0	3	CS[31-24] should be protected
CAN_1	ID22	0x1E4	3	ID Register of Message Buffer 22

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_1	CS23	0x1F0	3	CS[31-24] should be protected
CAN_1	ID23	0x1F4	3	ID Register of Message Buffer 23
CAN_1	CS24	0x200	3	CS[31-24] should be protected
CAN_1	ID24	0x204	3	ID Register of Message Buffer 24
CAN_1	CS25	0x210	3	CS[31-24] should be protected
CAN_1	ID25	0x214	3	ID Register of Message Buffer 25
CAN_1	CS26	0x220	3	CS[31-24] should be protected
CAN_1	ID26	0x224	3	ID Register of Message Buffer 26
CAN_1	CS27	0x230	3	CS[31-24] should be protected
CAN_1	ID27	0x234	3	ID Register of Message Buffer 27
CAN_1	CS28	0x240	3	CS[31-24] should be protected
CAN_1	ID28	0x244	3	ID Register of Message Buffer 28
CAN_1	CS29	0x250	3	CS[31-24] should be protected
CAN_1	ID29	0x254	3	ID Register of Message Buffer 29
CAN_1	CS30	0x260	3	CS[31-24] should be protected
CAN_1	ID30	0x264	3	ID Register of Message Buffer 30
CAN_1	CS31	0x270	3	CS[31-24] should be protected
CAN_1	ID31	0x274	3	ID Register of Message Buffer 31
CAN_1	CS32	0x280	3	CS[31-24] should be protected
CAN_1	ID32	0x284	3	ID Register of Message Buffer 32
CAN_1	CS33	0x290	3	CS[31-24] should be protected
CAN_1	ID33	0x294	3	ID Register of Message Buffer 33
CAN_1	CS34	0x2a0	3	CS[31-24] should be protected
CAN_1	ID34	0x2a4	3	ID Register of Message Buffer 34
CAN_1	CS35	0x2b0	3	CS[31-24] should be protected
CAN_1	ID35	0x2b4	3	ID Register of Message Buffer 35
CAN_1	CS36	0x2c0	3	CS[31-24] should be protected
CAN_1	ID36	0x2c4	3	ID Register of Message Buffer 36
CAN_1	CS37	0x2d0	3	CS[31-24] should be protected
CAN_1	ID37	0x2d4	3	ID Register of Message Buffer 37
CAN_1	CS38	0x2e0	3	CS[31-24] should be protected
CAN_1	ID38	0x2e4	3	ID Register of Message Buffer 38
CAN_1	CS39	0x2f0	3	CS[31-24] should be protected
CAN_1	ID39	0x2f4	3	ID Register of Message Buffer 39
CAN_1	CS40	0x300	3	CS[31-24] should be protected
CAN_1	ID40	0x304	3	ID Register of Message Buffer 40
CAN_1	CS41	0x310	3	CS[31-24] should be protected
CAN_1	ID41	0x314	3	ID Register of Message Buffer 41
CAN_1	CS42	0x320	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_1	ID42	0x324	3	ID Register of Message Buffer 42
CAN_1	CS43	0x330	3	CS[31-24] should be protected
CAN_1	ID43	0x334	3	ID Register of Message Buffer 43
CAN_1	CS44	0x340	3	CS[31-24] should be protected
CAN_1	ID44	0x344	3	ID Register of Message Buffer 44
CAN_1	CS45	0x350	3	CS[31-24] should be protected
CAN_1	ID45	0x354	3	ID Register of Message Buffer 45
CAN_1	CS46	0x360	3	CS[31-24] should be protected
CAN_1	ID46	0x364	3	ID Register of Message Buffer 46
CAN_1	CS47	0x370	3	CS[31-24] should be protected
CAN_1	ID47	0x374	3	ID Register of Message Buffer 47
CAN_1	CS48	0x380	3	CS[31-24] should be protected
CAN_1	ID48	0x384	3	ID Register of Message Buffer 48
CAN_1	CS49	0x390	3	CS[31-24] should be protected
CAN_1	ID49	0x394	3	ID Register of Message Buffer 49
CAN_1	CS50	0x3a0	3	CS[31-24] should be protected
CAN_1	ID50	0x3a4	3	ID Register of Message Buffer 50
CAN_1	CS51	0x3b0	3	CS[31-24] should be protected
CAN_1	ID51	0x3b4	3	ID Register of Message Buffer 51
CAN_1	CS52	0x3c0	3	CS[31-24] should be protected
CAN_1	ID52	0x3c4	3	ID Register of Message Buffer 52
CAN_1	CS53	0x3d0	3	CS[31-24] should be protected
CAN_1	ID53	0x3d4	3	ID Register of Message Buffer 53
CAN_1	CS54	0x3e0	3	CS[31-24] should be protected
CAN_1	ID54	0x3e4	3	ID Register of Message Buffer 54
CAN_1	CS55	0x3f0	3	CS[31-24] should be protected
CAN_1	ID55	0x3f4	3	ID Register of Message Buffer 55
CAN_1	CS56	0x400	3	CS[31-24] should be protected
CAN_1	ID56	0x404	3	ID Register of Message Buffer 56
CAN_1	CS57	0x410	3	CS[31-24] should be protected
CAN_1	ID57	0x414	3	ID Register of Message Buffer 57
CAN_1	CS58	0x420	3	CS[31-24] should be protected
CAN_1	ID58	0x424	3	ID Register of Message Buffer 58
CAN_1	CS59	0x430	3	CS[31-24] should be protected
CAN_1	ID59	0x434	3	ID Register of Message Buffer 59
CAN_1	CS60	0x440	3	CS[31-24] should be protected
CAN_1	ID60	0x444	3	ID Register of Message Buffer 60
CAN_1	CS61	0x450	3	CS[31-24] should be protected
CAN_1	ID61	0x454	3	ID Register of Message Buffer 61

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_1	CS62	0x460	3	CS[31-24] should be protected
CAN_1	ID62	0x464	3	ID Register of Message Buffer 62
CAN_1	CS63	0x470	3	CS[31-24] should be protected
CAN_1	ID63	0x474	3	ID Register of Message Buffer 63
CAN_1	RXIMR0	0x880	3	Rx Individual Mask Register 0
CAN_1	RXIMR1	0x884	3	Rx Individual Mask Register 1
CAN_1	RXIMR2	0x888	3	Rx Individual Mask Register 2
CAN_1	RXIMR3	0x88C	3	Rx Individual Mask Register 3
CAN_1	RXIMR4	0x890	3	Rx Individual Mask Register 4
CAN_1	RXIMR5	0x894	3	Rx Individual Mask Register 5
CAN_1	RXIMR6	0x898	3	Rx Individual Mask Register 6
CAN_1	RXIMR7	0x89C	3	Rx Individual Mask Register 7
CAN_1	RXIMR8	0x8A0	3	Rx Individual Mask Register 8
CAN_1	RXIMR9	0x8A4	3	Rx Individual Mask Register 9
CAN_1	RXIMR10	0x8A8	3	Rx Individual Mask Register 10
CAN_1	RXIMR11	0x8AC	3	Rx Individual Mask Register 11
CAN_1	RXIMR12	0x8B0	3	Rx Individual Mask Register 12
CAN_1	RXIMR13	0x8B4	3	Rx Individual Mask Register 13
CAN_1	RXIMR14	0x8B8	3	Rx Individual Mask Register 14
CAN_1	RXIMR15	0x8BC	3	Rx Individual Mask Register 15
CAN_1	RXIMR16	0x8C0	3	Rx Individual Mask Register 16
CAN_1	RXIMR17	0x8C4	3	Rx Individual Mask Register 17
CAN_1	RXIMR18	0x8C8	3	Rx Individual Mask Register 18
CAN_1	RXIMR19	0x8CC	3	Rx Individual Mask Register 19
CAN_1	RXIMR20	0x8D0	3	Rx Individual Mask Register 20
CAN_1	RXIMR21	0x8D4	3	Rx Individual Mask Register 21
CAN_1	RXIMR22	0x8D8	3	Rx Individual Mask Register 22
CAN_1	RXIMR23	0x8DC	3	Rx Individual Mask Register 23
CAN_1	RXIMR24	0x8E0	3	Rx Individual Mask Register 24
CAN_1	RXIMR25	0x8E4	3	Rx Individual Mask Register 25
CAN_1	RXIMR26	0x8E8	3	Rx Individual Mask Register 26
CAN_1	RXIMR27	0x8EC	3	Rx Individual Mask Register 27
CAN_1	RXIMR28	0x8F0	3	Rx Individual Mask Register 28
CAN_1	RXIMR29	0x8F4	3	Rx Individual Mask Register 29
CAN_1	RXIMR30	0x8F8	3	Rx Individual Mask Register 30
CAN_1	RXIMR31	0x8FC	3	Rx Individual Mask Register 31
CAN_1	RXIMR32	0x900	3	Rx Individual Mask Register 32
CAN_1	RXIMR33	0x904	3	Rx Individual Mask Register 33
CAN_1	RXIMR34	0x908	3	Rx Individual Mask Register 34

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_1	RXIMR35	0x90C	3	Rx Individual Mask Register 35
CAN_1	RXIMR36	0x910	3	Rx Individual Mask Register 36
CAN_1	RXIMR37	0x914	3	Rx Individual Mask Register 37
CAN_1	RXIMR38	0x918	3	Rx Individual Mask Register 38
CAN_1	RXIMR39	0x91C	3	Rx Individual Mask Register 39
CAN_1	RXIMR40	0x920	3	Rx Individual Mask Register 40
CAN_1	RXIMR41	0x924	3	Rx Individual Mask Register 41
CAN_1	RXIMR42	0x928	3	Rx Individual Mask Register 42
CAN_1	RXIMR43	0x92C	3	Rx Individual Mask Register 43
CAN_1	RXIMR44	0x930	3	Rx Individual Mask Register 44
CAN_1	RXIMR45	0x934	3	Rx Individual Mask Register 45
CAN_1	RXIMR46	0x938	3	Rx Individual Mask Register 46
CAN_1	RXIMR47	0x93C	3	Rx Individual Mask Register 47
CAN_1	RXIMR48	0x940	3	Rx Individual Mask Register 48
CAN_1	RXIMR49	0x944	3	Rx Individual Mask Register 49
CAN_1	RXIMR50	0x948	3	Rx Individual Mask Register 50
CAN_1	RXIMR51	0x94C	3	Rx Individual Mask Register 51
CAN_1	RXIMR52	0x950	3	Rx Individual Mask Register 52
CAN_1	RXIMR53	0x954	3	Rx Individual Mask Register 53
CAN_1	RXIMR54	0x958	3	Rx Individual Mask Register 54
CAN_1	RXIMR55	0x95C	3	Rx Individual Mask Register 55
CAN_1	RXIMR56	0x960	3	Rx Individual Mask Register 56
CAN_1	RXIMR57	0x964	3	Rx Individual Mask Register 57
CAN_1	RXIMR58	0x968	3	Rx Individual Mask Register 58
CAN_1	RXIMR59	0x96C	3	Rx Individual Mask Register 59
CAN_1	RXIMR60	0x970	3	Rx Individual Mask Register 60
CAN_1	RXIMR61	0x974	3	Rx Individual Mask Register 61
CAN_1	RXIMR62	0x978	3	Rx Individual Mask Register 62
CAN_1	RXIMR63	0x97C	3	Rx Individual Mask Register 63
CAN_1	MECR	0xAE0	3	Memory error configuration register
CAN_2	MCR	0x0	3	Module Configuration
CAN_2	CTRL1	0x4	3	Control Register
CAN_2	RXMGMASK	0x10	3	Rx Global Mask
CAN_2	RX14MASK	0x14	3	Rx Buffer 14 Mask
CAN_2	RX15MASK	0x18	3	Rx Buffer 15 Mask
CAN_2	IMASK2	0x24	3	Interrupt Masks 2
CAN_2	IMASK1	0x28	3	Interrupt Masks 1
CAN_2	CTRL2	0x34	3	Control 2
CAN_2	CS0	0x80	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	ID0	0x84	3	ID Register of Message Buffer 0
CAN_2	CS1	0x90	3	CS[31-24] should be protected
CAN_2	ID1	0x94	3	ID Register of Message Buffer 1
CAN_2	CS2	0xA0	3	CS[31-24] should be protected
CAN_2	ID2	0xA4	3	ID Register of Message Buffer 2
CAN_2	CS3	0xB0	3	CS[31-24] should be protected
CAN_2	ID3	0xB4	3	ID Register of Message Buffer 3
CAN_2	CS4	0xC0	3	CS[31-24] should be protected
CAN_2	ID4	0xC4	3	ID Register of Message Buffer 4
CAN_2	CS5	0xD0	3	CS[31-24] should be protected
CAN_2	ID5	0xD4	3	ID Register of Message Buffer 5
CAN_2	CS6	0xE0	3	CS[31-24] should be protected
CAN_2	ID6	0xE4	3	ID Register of Message Buffer 6
CAN_2	CS7	0xF0	3	CS[31-24] should be protected
CAN_2	ID7	0xF4	3	ID Register of Message Buffer 7
CAN_2	CS8	0x100	3	CS[31-24] should be protected
CAN_2	ID8	0x104	3	ID Register of Message Buffer 8
CAN_2	CS9	0x110	3	CS[31-24] should be protected
CAN_2	ID9	0x114	3	ID Register of Message Buffer 9
CAN_2	CS10	0x120	3	CS[31-24] should be protected
CAN_2	ID10	0x124	3	ID Register of Message Buffer 10
CAN_2	CS11	0x130	3	CS[31-24] should be protected
CAN_2	ID11	0x134	3	ID Register of Message Buffer 11
CAN_2	CS12	0x140	3	CS[31-24] should be protected
CAN_2	ID12	0x144	3	ID Register of Message Buffer 12
CAN_2	CS13	0x150	3	CS[31-24] should be protected
CAN_2	ID13	0x154	3	ID Register of Message Buffer 13
CAN_2	CS14	0x160	3	CS[31-24] should be protected
CAN_2	ID14	0x164	3	ID Register of Message Buffer 14
CAN_2	CS15	0x170	3	CS[31-24] should be protected
CAN_2	ID15	0x174	3	ID Register of Message Buffer 15
CAN_2	CS16	0x180	3	CS[31-24] should be protected
CAN_2	ID16	0x184	3	ID Register of Message Buffer 16
CAN_2	CS17	0x190	3	CS[31-24] should be protected
CAN_2	ID17	0x194	3	ID Register of Message Buffer 17
CAN_2	CS18	0x1A0	3	CS[31-24] should be protected
CAN_2	ID18	0x1A4	3	ID Register of Message Buffer 18
CAN_2	CS19	0x1B0	3	CS[31-24] should be protected
CAN_2	ID19	0x1B4	3	ID Register of Message Buffer 19

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	CS20	0x1C0	3	CS[31-24] should be protected
CAN_2	ID20	0x1C4	3	ID Register of Message Buffer 20
CAN_2	CS21	0x1D0	3	CS[31-24] should be protected
CAN_2	ID21	0x1D4	3	ID Register of Message Buffer 21
CAN_2	CS22	0x1E0	3	CS[31-24] should be protected
CAN_2	ID22	0x1E4	3	ID Register of Message Buffer 22
CAN_2	CS23	0x1F0	3	CS[31-24] should be protected
CAN_2	ID23	0x1F4	3	ID Register of Message Buffer 23
CAN_2	CS24	0x200	3	CS[31-24] should be protected
CAN_2	ID24	0x204	3	ID Register of Message Buffer 24
CAN_2	CS25	0x210	3	CS[31-24] should be protected
CAN_2	ID25	0x214	3	ID Register of Message Buffer 25
CAN_2	CS26	0x220	3	CS[31-24] should be protected
CAN_2	ID26	0x224	3	ID Register of Message Buffer 26
CAN_2	CS27	0x230	3	CS[31-24] should be protected
CAN_2	ID27	0x234	3	ID Register of Message Buffer 27
CAN_2	CS28	0x240	3	CS[31-24] should be protected
CAN_2	ID28	0x244	3	ID Register of Message Buffer 28
CAN_2	CS29	0x250	3	CS[31-24] should be protected
CAN_2	ID29	0x254	3	ID Register of Message Buffer 29
CAN_2	CS30	0x260	3	CS[31-24] should be protected
CAN_2	ID30	0x264	3	ID Register of Message Buffer 30
CAN_2	CS31	0x270	3	CS[31-24] should be protected
CAN_2	ID31	0x274	3	ID Register of Message Buffer 31
CAN_2	CS32	0x280	3	CS[31-24] should be protected
CAN_2	ID32	0x284	3	ID Register of Message Buffer 32
CAN_2	CS33	0x290	3	CS[31-24] should be protected
CAN_2	ID33	0x294	3	ID Register of Message Buffer 33
CAN_2	CS34	0x2a0	3	CS[31-24] should be protected
CAN_2	ID34	0x2a4	3	ID Register of Message Buffer 34
CAN_2	CS35	0x2b0	3	CS[31-24] should be protected
CAN_2	ID35	0x2b4	3	ID Register of Message Buffer 35
CAN_2	CS36	0x2c0	3	CS[31-24] should be protected
CAN_2	ID36	0x2c4	3	ID Register of Message Buffer 36
CAN_2	CS37	0x2d0	3	CS[31-24] should be protected
CAN_2	ID37	0x2d4	3	ID Register of Message Buffer 37
CAN_2	CS38	0x2E0	3	CS[31-24] should be protected
CAN_2	ID38	0x2E4	3	ID Register of Message Buffer 38
CAN_2	CS39	0x2f0	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	ID39	0x2f4	3	ID Register of Message Buffer 39
CAN_2	CS40	0x300	3	CS[31-24] should be protected
CAN_2	ID40	0x304	3	ID Register of Message Buffer 40
CAN_2	CS41	0x310	3	CS[31-24] should be protected
CAN_2	ID41	0x314	3	ID Register of Message Buffer 41
CAN_2	CS42	0x320	3	CS[31-24] should be protected
CAN_2	ID42	0x324	3	ID Register of Message Buffer 42
CAN_2	CS43	0x330	3	CS[31-24] should be protected
CAN_2	ID43	0x334	3	ID Register of Message Buffer 43
CAN_2	CS44	0x340	3	CS[31-24] should be protected
CAN_2	ID44	0x344	3	ID Register of Message Buffer 44
CAN_2	CS45	0x350	3	CS[31-24] should be protected
CAN_2	ID45	0x354	3	ID Register of Message Buffer 45
CAN_2	CS46	0x360	3	CS[31-24] should be protected
CAN_2	ID46	0x364	3	ID Register of Message Buffer 46
CAN_2	CS47	0x370	3	CS[31-24] should be protected
CAN_2	ID47	0x374	3	ID Register of Message Buffer 47
CAN_2	CS48	0x380	3	CS[31-24] should be protected
CAN_2	ID48	0x384	3	ID Register of Message Buffer 48
CAN_2	CS49	0x390	3	CS[31-24] should be protected
CAN_2	ID49	0x394	3	ID Register of Message Buffer 49
CAN_2	CS50	0x3a0	3	CS[31-24] should be protected
CAN_2	ID50	0x3a4	3	ID Register of Message Buffer 50
CAN_2	CS51	0x3b0	3	CS[31-24] should be protected
CAN_2	ID51	0x3b4	3	ID Register of Message Buffer 51
CAN_2	CS52	0x3c0	3	CS[31-24] should be protected
CAN_2	ID52	0x3c4	3	ID Register of Message Buffer 52
CAN_2	CS53	0x3d0	3	CS[31-24] should be protected
CAN_2	ID53	0x3d4	3	ID Register of Message Buffer 53
CAN_2	CS54	0x3E0	3	CS[31-24] should be protected
CAN_2	ID54	0x3E4	3	ID Register of Message Buffer 54
CAN_2	CS55	0x3f0	3	CS[31-24] should be protected
CAN_2	ID55	0x3f4	3	ID Register of Message Buffer 55
CAN_2	CS56	0x400	3	CS[31-24] should be protected
CAN_2	ID56	0x404	3	ID Register of Message Buffer 56
CAN_2	CS57	0x410	3	CS[31-24] should be protected
CAN_2	ID57	0x414	3	ID Register of Message Buffer 57
CAN_2	CS58	0x420	3	CS[31-24] should be protected
CAN_2	ID58	0x424	3	ID Register of Message Buffer 58

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	CS59	0x430	3	CS[31-24] should be protected
CAN_2	ID59	0x434	3	ID Register of Message Buffer 59
CAN_2	CS60	0x440	3	CS[31-24] should be protected
CAN_2	ID60	0x444	3	ID Register of Message Buffer 60
CAN_2	CS61	0x450	3	CS[31-24] should be protected
CAN_2	ID61	0x454	3	ID Register of Message Buffer 61
CAN_2	CS62	0x460	3	CS[31-24] should be protected
CAN_2	ID62	0x464	3	ID Register of Message Buffer 62
CAN_2	CS63	0x470	3	CS[31-24] should be protected
CAN_2	ID63	0x474	3	ID Register of Message Buffer 63
CAN_2	CS64	0x480	3	CS[31-24] should be protected
CAN_2	ID64	0x484	3	ID Register of Message Buffer 64
CAN_2	CS65	0x490	3	CS[31-24] should be protected
CAN_2	ID65	0x494	3	ID Register of Message Buffer 65
CAN_2	CS66	0x4A0	3	CS[31-24] should be protected
CAN_2	ID66	0x4A4	3	ID Register of Message Buffer 66
CAN_2	CS67	0x4B0	3	CS[31-24] should be protected
CAN_2	ID67	0x4B4	3	ID Register of Message Buffer 67
CAN_2	ID68	0x4C4	3	ID Register of Message Buffer 68
CAN_2	CS69	0x4D0	3	CS[31-24] should be protected
CAN_2	ID69	0x4D4	3	ID Register of Message Buffer 69
CAN_2	ID70	0x4E4	3	ID Register of Message Buffer 70
CAN_2	CS71	0x4F0	3	CS[31-24] should be protected
CAN_2	ID71	0x4F4	3	ID Register of Message Buffer 71
CAN_2	CS72	0x500	3	CS[31-24] should be protected
CAN_2	ID72	0x504	3	ID Register of Message Buffer 72
CAN_2	CS73	0x510	3	CS[31-24] should be protected
CAN_2	ID73	0x514	3	ID Register of Message Buffer 73
CAN_2	CS74	0x520	3	CS[31-24] should be protected
CAN_2	ID74	0x524	3	ID Register of Message Buffer 74
CAN_2	CS75	0x530	3	CS[31-24] should be protected
CAN_2	ID75	0x534	3	ID Register of Message Buffer 75
CAN_2	CS76	0x540	3	CS[31-24] should be protected
CAN_2	ID76	0x544	3	ID Register of Message Buffer 76
CAN_2	CS77	0x550	3	CS[31-24] should be protected
CAN_2	ID77	0x554	3	ID Register of Message Buffer 77
CAN_2	CS78	0x560	3	CS[31-24] should be protected
CAN_2	ID78	0x564	3	ID Register of Message Buffer 78
CAN_2	CS79	0x570	3	CS[31-24] should be protected

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	ID79	0x574	3	ID Register of Message Buffer 79
CAN_2	CS80	0x580	3	CS[31-24] should be protected
CAN_2	ID80	0x584	3	ID Register of Message Buffer 80
CAN_2	CS81	0x590	3	CS[31-24] should be protected
CAN_2	ID81	0x594	3	ID Register of Message Buffer 81
CAN_2	CS82	0x5A0	3	CS[31-24] should be protected
CAN_2	ID82	0x5A4	3	ID Register of Message Buffer 82
CAN_2	CS83	0x5B0	3	CS[31-24] should be protected
CAN_2	ID83	0x5B4	3	ID Register of Message Buffer 83
CAN_2	CS84	0x5C0	3	CS[31-24] should be protected
CAN_2	ID84	0x5C4	3	ID Register of Message Buffer 84
CAN_2	CS85	0x5D0	3	CS[31-24] should be protected
CAN_2	ID85	0x5D4	3	ID Register of Message Buffer 85
CAN_2	CS86	0x5E0	3	CS[31-24] should be protected
CAN_2	ID86	0x5E4	3	ID Register of Message Buffer 86
CAN_2	CS88	0x600	3	CS[31-24] should be protected
CAN_2	ID88	0x604	3	ID Register of Message Buffer 87
CAN_2	RXIMR0	0x880	3	Rx Individual Mask Register 0
CAN_2	RXIMR1	0x884	3	Rx Individual Mask Register 1
CAN_2	RXIMR2	0x888	3	Rx Individual Mask Register 2
CAN_2	RXIMR3	0x88C	3	Rx Individual Mask Register 3
CAN_2	RXIMR4	0x890	3	Rx Individual Mask Register 4
CAN_2	RXIMR5	0x894	3	Rx Individual Mask Register 5
CAN_2	RXIMR6	0x898	3	Rx Individual Mask Register 6
CAN_2	RXIMR7	0x89C	3	Rx Individual Mask Register 7
CAN_2	RXIMR8	0x8A0	3	Rx Individual Mask Register 8
CAN_2	RXIMR9	0x8A4	3	Rx Individual Mask Register 9
CAN_2	RXIMR10	0x8A8	3	Rx Individual Mask Register 10
CAN_2	RXIMR11	0x8AC	3	Rx Individual Mask Register 11
CAN_2	RXIMR12	0x8B0	3	Rx Individual Mask Register 12
CAN_2	RXIMR13	0x8B4	3	Rx Individual Mask Register 13
CAN_2	RXIMR14	0x8B8	3	Rx Individual Mask Register 14
CAN_2	RXIMR15	0x8BC	3	Rx Individual Mask Register 15
CAN_2	RXIMR16	0x8C0	3	Rx Individual Mask Register 16
CAN_2	RXIMR17	0x8C4	3	Rx Individual Mask Register 17
CAN_2	RXIMR18	0x8C8	3	Rx Individual Mask Register 18
CAN_2	RXIMR19	0x8CC	3	Rx Individual Mask Register 19
CAN_2	RXIMR20	0x8D0	3	Rx Individual Mask Register 20
CAN_2	RXIMR21	0x8D4	3	Rx Individual Mask Register 21

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	RXIMR22	0x8D8	3	Rx Individual Mask Register 22
CAN_2	RXIMR23	0x8DC	3	Rx Individual Mask Register 23
CAN_2	RXIMR24	0x8E0	3	Rx Individual Mask Register 24
CAN_2	RXIMR25	0x8E4	3	Rx Individual Mask Register 25
CAN_2	RXIMR26	0x8E8	3	Rx Individual Mask Register 26
CAN_2	RXIMR27	0x8EC	3	Rx Individual Mask Register 27
CAN_2	RXIMR28	0x8F0	3	Rx Individual Mask Register 28
CAN_2	RXIMR29	0x8F4	3	Rx Individual Mask Register 29
CAN_2	RXIMR30	0x8F8	3	Rx Individual Mask Register 30
CAN_2	RXIMR31	0x8FC	3	Rx Individual Mask Register 31
CAN_2	RXIMR32	0x900	3	Rx Individual Mask Register 32
CAN_2	RXIMR33	0x904	3	Rx Individual Mask Register 33
CAN_2	RXIMR34	0x908	3	Rx Individual Mask Register 34
CAN_2	RXIMR35	0x90C	3	Rx Individual Mask Register 35
CAN_2	RXIMR36	0x910	3	Rx Individual Mask Register 36
CAN_2	RXIMR37	0x914	3	Rx Individual Mask Register 37
CAN_2	RXIMR38	0x918	3	Rx Individual Mask Register 38
CAN_2	RXIMR39	0x91C	3	Rx Individual Mask Register 39
CAN_2	RXIMR40	0x920	3	Rx Individual Mask Register 40
CAN_2	RXIMR41	0x924	3	Rx Individual Mask Register 41
CAN_2	RXIMR42	0x928	3	Rx Individual Mask Register 42
CAN_2	RXIMR43	0x92C	3	Rx Individual Mask Register 43
CAN_2	RXIMR44	0x930	3	Rx Individual Mask Register 44
CAN_2	RXIMR45	0x934	3	Rx Individual Mask Register 45
CAN_2	RXIMR46	0x938	3	Rx Individual Mask Register 46
CAN_2	RXIMR47	0x93C	3	Rx Individual Mask Register 47
CAN_2	RXIMR48	0x940	3	Rx Individual Mask Register 48
CAN_2	RXIMR49	0x944	3	Rx Individual Mask Register 49
CAN_2	RXIMR50	0x948	3	Rx Individual Mask Register 50
CAN_2	RXIMR51	0x94C	3	Rx Individual Mask Register 51
CAN_2	RXIMR52	0x950	3	Rx Individual Mask Register 52
CAN_2	RXIMR53	0x954	3	Rx Individual Mask Register 53
CAN_2	RXIMR54	0x958	3	Rx Individual Mask Register 54
CAN_2	RXIMR55	0x95C	3	Rx Individual Mask Register 55
CAN_2	RXIMR56	0x960	3	Rx Individual Mask Register 56
CAN_2	RXIMR57	0x964	3	Rx Individual Mask Register 57
CAN_2	RXIMR58	0x968	3	Rx Individual Mask Register 58
CAN_2	RXIMR59	0x96C	3	Rx Individual Mask Register 59
CAN_2	RXIMR60	0x970	3	Rx Individual Mask Register 60

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CAN_2	RXIMR61	0x974	3	Rx Individual Mask Register 61
CAN_2	RXIMR62	0x978	3	Rx Individual Mask Register 62
CAN_2	RXIMR63	0x97C	3	Rx Individual Mask Register 63
CAN_2	RXIMR64	0x980	3	Rx Individual Mask Register 64
CAN_2	RXIMR65	0x984	3	Rx Individual Mask Register 65
CAN_2	RXIMR66	0x988	3	Rx Individual Mask Register 66
CAN_2	RXIMR67	0x98C	3	Rx Individual Mask Register 67
CAN_2	RXIMR68	0x990	3	Rx Individual Mask Register 68
CAN_2	RXIMR69	0x994	3	Rx Individual Mask Register 69
CAN_2	RXIMR70	0x998	3	Rx Individual Mask Register 70
CAN_2	RXIMR71	0x99C	3	Rx Individual Mask Register 71
CAN_2	RXIMR72	0x9A0	3	Rx Individual Mask Register 72
CAN_2	RXIMR73	0x9A4	3	Rx Individual Mask Register 73
CAN_2	RXIMR74	0x9A8	3	Rx Individual Mask Register 74
CAN_2	RXIMR75	0x9AC	3	Rx Individual Mask Register 75
CAN_2	RXIMR76	0x9B0	3	Rx Individual Mask Register 76
CAN_2	RXIMR77	0x9B4	3	Rx Individual Mask Register 77
CAN_2	RXIMR78	0x9B8	3	Rx Individual Mask Register 78
CAN_2	RXIMR79	0x9BC	3	Rx Individual Mask Register 79
CAN_2	RXIMR80	0x9C0	3	Rx Individual Mask Register 80
CAN_2	RXIMR81	0x9C4	3	Rx Individual Mask Register 81
CAN_2	RXIMR82	0x9C8	3	Rx Individual Mask Register 82
CAN_2	RXIMR83	0x9CC	3	Rx Individual Mask Register 83
CAN_2	RXIMR84	0x9D0	3	Rx Individual Mask Register 84
CAN_2	RXIMR85	0x9D4	3	Rx Individual Mask Register 85
CAN_2	RXIMR86	0x9D8	3	Rx Individual Mask Register 86
CAN_2	RXIMR87	0x9DC	3	Rx Individual Mask Register 87
CAN_2	RXIMR88	0x9E0	3	Rx Individual Mask Register 88
CAN_2	RXIMR89	0x9E4	3	Rx Individual Mask Register 89
CAN_2	RXIMR90	0x9E8	3	Rx Individual Mask Register 90
CAN_2	RXIMR91	0x9EC	3	Rx Individual Mask Register 91
CAN_2	RXIMR92	0x9F0	3	Rx Individual Mask Register 92
CAN_2	RXIMR93	0x9F4	3	Rx Individual Mask Register 93
CAN_2	RXIMR94	0x9F8	3	Rx Individual Mask Register 94
CAN_2	RXIMR95	0x9FC	3	Rx Individual Mask Register 95
CAN_2	MECR	0xAE0	3	Memory error configuration register
CAN_2	FDCTRL	0xC00	3	CAN FD Control Register
CAN_2	FDCBT	0xC04	3	CAN FD Bit Timing Register
FlexPWM_0	SUB0_INIT	0x02	2	Initial Count Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FlexPWM_0	SUB0_CTRL2	0x04	2	Control 2 Register
FlexPWM_0	SUB0_CTRL1	0x06	2	Control Register
FlexPWM_0	SUB0_VAL0	0x08	2	Value Register 0
FlexPWM_0	SUB0_VAL1	0x0A	2	Value Register 1
FlexPWM_0	SUB0_VAL2	0x0C	2	Value Register 2
FlexPWM_0	SUB0_VAL3	0x0E	2	Value Register 3
FlexPWM_0	SUB0_VAL4	0x10	2	Value Register 4
FlexPWM_0	SUB0_VAL5	0x12	2	Value Register 5
FlexPWM_0	SUB0_OCTRL	0x18	2	Output Control Register
FlexPWM_0	SUB0_INTEN	0x1C	2	Interrupt Enable Register
FlexPWM_0	SUB0_DMAEN	0x1E	2	DMA Enable Register
FlexPWM_0	SUB0_TCTRL	0x20	2	Output Trigger Control Register
FlexPWM_0	SUB0_DISMAP	0x22	2	Fault Disable Mapping Register
FlexPWM_0	SUB0_DTCNT0	0x24	2	Deadtime Count Register 0
FlexPWM_0	SUB0_DTCNT1	0x26	2	Deadtime Count Register 1
FlexPWM_0	SUB0_CAPTCTRLX	0x30	2	Capture Control Register X
FlexPWM_0	SUB0_CAPTCMPX	0x32	2	Capture Compare Register X
FlexPWM_0	SUB1_INIT	0x52	2	Initial Count Register
FlexPWM_0	SUB1_CTRL2	0x54	2	Control 2 Register
FlexPWM_0	SUB1_CTRL1	0x56	2	Control Register
FlexPWM_0	SUB1_VAL0	0x58	2	Value Register 0
FlexPWM_0	SUB1_VAL1	0x5A	2	Value Register 1
FlexPWM_0	SUB1_VAL2	0x5C	2	Value Register 2
FlexPWM_0	SUB1_VAL3	0x5E	2	Value Register 3
FlexPWM_0	SUB1_VAL4	0x60	2	Value Register 4
FlexPWM_0	SUB1_VAL5	0x62	2	Value Register 5
FlexPWM_0	SUB1_OCTRL	0x68	2	Output Control Register
FlexPWM_0	SUB1_INTEN	0x6C	2	Interrupt Enable Register
FlexPWM_0	SUB1_DMAEN	0x6E	2	DMA Enable Register
FlexPWM_0	SUB1_TCTRL	0x70	2	Output Trigger Control Register
FlexPWM_0	SUB1_DISMAP	0x72	2	Fault Disable Mapping Register
FlexPWM_0	SUB1_DTCNT0	0x74	2	Deadtime Count Register 0
FlexPWM_0	SUB1_DTCNT1	0x76	2	Deadtime Count Register 1
FlexPWM_0	SUB1_CAPTCTRLX	0x80	2	Capture Control Register X
FlexPWM_0	SUB1_CAPTCMPX	0x82	2	Capture Compare Register X
FlexPWM_0	SUB2_INIT	0xA2	2	Initial Count Register
FlexPWM_0	SUB2_CTRL2	0xA4	2	Control 2 Register
FlexPWM_0	SUB2_CTRL1	0xA6	2	Control Register
FlexPWM_0	SUB2_VAL0	0xA8	2	Value Register 0

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FlexPWM_0	SUB2_VAL1	0xAA	2	Value Register 1
FlexPWM_0	SUB2_VAL2	0xAC	2	Value Register 2
FlexPWM_0	SUB2_VAL3	0xAE	2	Value Register 3
FlexPWM_0	SUB2_VAL4	0xB0	2	Value Register 4
FlexPWM_0	SUB2_VAL5	0xB2	2	Value Register 5
FlexPWM_0	SUB2_OCTRL	0xB8	2	Output Control Register
FlexPWM_0	SUB2_INTEN	0xBC	2	Interrupt Enable Register
FlexPWM_0	SUB2_DMAEN	0xBE	2	DMA Enable Register
FlexPWM_0	SUB2_TCTRL	0xC0	2	Output Trigger Control Register
FlexPWM_0	SUB2_DISMAP	0xC2	2	Fault Disable Mapping Register
FlexPWM_0	SUB2_DTCNT0	0xC4	2	Deadtime Count Register 0
FlexPWM_0	SUB2_DTCNT1	0xC6	2	Deadtime Count Register 1
FlexPWM_0	SUB2_CAPTCTRLX	0xD0	2	Capture Control Register X
FlexPWM_0	SUB2_CAPTCMPX	0xD2	2	Capture Compare Register X
FlexPWM_0	SUB3_INIT	0xF2	2	Initial Count Register
FlexPWM_0	SUB3_CTRL2	0xF4	2	Control 2 Register
FlexPWM_0	SUB3_CTRL1	0xF6	2	Control Register
FlexPWM_0	SUB3_VAL0	0xF8	2	Value Register 0
FlexPWM_0	SUB3_VAL1	0xFA	2	Value Register 1
FlexPWM_0	SUB3_VAL2	0xFC	2	Value Register 2
FlexPWM_0	SUB3_VAL3	0xFE	2	Value Register 3
FlexPWM_0	SUB3_VAL4	0x100	2	Value Register 4
FlexPWM_0	SUB3_VAL5	0x102	2	Value Register 5
FlexPWM_0	SUB3_OCTRL	0x108	2	Output Control Register
FlexPWM_0	SUB3_INTEN	0x10C	2	Interrupt Enable Register
FlexPWM_0	SUB3_DMAEN	0x10E	2	DMA Enable Register
FlexPWM_0	SUB3_TCTRL	0x110	2	Output Trigger Control Register
FlexPWM_0	SUB3_DISMAP	0x112	2	Fault Disable Mapping Register
FlexPWM_0	SUB3_DTCNT0	0x114	2	Deadtime Count Register 0
FlexPWM_0	SUB3_DTCNT1	0x116	2	Deadtime Count Register 1
FlexPWM_0	SUB3_CAPTCTRLX	0x120	2	Capture Control Register X
FlexPWM_0	SUB3_CAPTCMPX	0x122	2	Capture Compare Register X
FlexPWM_0	OUTEN	0x140	2	Output Enable Register
FlexPWM_0	MASK	0x142	2	Output Mask Register
FlexPWM_0	SWCOUT	0x144	2	Software Controlled Output Register
FlexPWM_0	DTSRCSEL	0x146	2	Deadtime Source Select Register
FlexPWM_0	MCTRL	0x148	2	Master Control Register
FlexPWM_0	FCTRL	0x14C	2	Fault Control Register
FR	MCR	0x02	2	configuration register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	SYMBADHR	0x04	2	base address register H
FR	SYMBADLR	0x06	2	base address register L
FR	STBSCR	0x08	2	strobe control
FR	MBDSR	0x0C	2	data size definition
FR	MBSSUTR	0x0E	2	message buffer configuration
FR	POCR	0x14	2	protocol operation control
FR	GIFER	0x16	2	global interrupt enable (only GIFER[8-15] should be protected)
FR	PIER0	0x1C	2	protocol interrupt enable
FR	PIER1	0x1E	2	protocol interrupt enable
FR	SYMATOR	0x3E	2	maximum time out
FR	SFTOR	0x42	2	memory offset
FR	SFTCCSR	0x44	2	sync frame table configuration
FR	SFIDRFR	0x46	2	sync frame ID rejection filter
FR	SFIDAFVR	0x48	2	sync frame ID acceptance filter
FR	SFIDAFMR	0x4A	2	sync frame ID mask filter
FR	NMVLR	0x58	2	protocol parameter
FR	TICCR	0x5A	2	timer configuration and control
FR	TI1CYSR	0x5C	2	Timer 1 Cycle Set Register
FR	TI1MTOR	0x5E	2	macrotik offset
FR	TI2CR0_ABS	0x60	2	timer2 configuration register
FR	TI2CR1_ABS	0x62	2	timer2 configuration register
FR	MTSACFR	0x80	2	MTS A configuration
FR	MTSBCFR	0x82	2	MTS B configuration
FR	RFRFCTR	0x9A	2	receive FIFO control register
FR	PCR0	0xA0	2	protocol configuration register
FR	PCR1	0xA2	2	protocol configuration register
FR	PCR2	0xA4	2	protocol configuration register
FR	PCR3	0xA6	2	protocol configuration register
FR	PCR4	0xA8	2	protocol configuration register
FR	PCR5	0xAA	2	protocol configuration register
FR	PCR6	0xAC	2	protocol configuration register
FR	PCR7	0xAE	2	protocol configuration register
FR	PCR8	0xB0	2	protocol configuration register
FR	PCR9	0xB2	2	protocol configuration register
FR	PCR10	0xB4	2	protocol configuration register
FR	PCR11	0xB6	2	protocol configuration register
FR	PCR12	0xB8	2	protocol configuration register
FR	PCR13	0xBA	2	protocol configuration register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	PCR14	0xBC	2	protocol configuration register
FR	PCR15	0xBE	2	protocol configuration register
FR	PCR16	0xC0	2	protocol configuration register
FR	PCR17	0xC2	2	protocol configuration register
FR	PCR18	0xC4	2	protocol configuration register
FR	PCR19	0xC6	2	protocol configuration register
FR	PCR20	0xC8	2	protocol configuration register
FR	PCR21	0xCA	2	protocol configuration register
FR	PCR22	0xCC	2	protocol configuration register
FR	PCR23	0xCE	2	protocol configuration register
FR	PCR24	0xD0	2	protocol configuration register
FR	PCR25	0xD2	2	protocol configuration register
FR	PCR26	0xD4	2	protocol configuration register
FR	PCR27	0xD6	2	protocol configuration register
FR	PCR28	0xD8	2	protocol configuration register
FR	PCR29	0xDA	2	protocol configuration register
FR	PCR30	0xDC	2	protocol configuration register
FR	MBCCFR0	0x802	2	protocol configuration register
FR	MBFIDR0	0x804	2	Frame ID Registers
FR	MBCCFR1	0x80a	2	message buffer configuration
FR	MBFIDR1	0x80c	2	Frame ID Registers
FR	MBCCFR2	0x812	2	message buffer configuration
FR	MBFIDR2	0x814	2	Frame ID Registers
FR	MBCCFR3	0x81a	2	message buffer configuration
FR	MBFIDR3	0x81c	2	Frame ID Registers
FR	MBCCFR4	0x822	2	message buffer configuration
FR	MBFIDR4	0x824	2	Frame ID Registers
FR	MBCCFR5	0x82a	2	message buffer configuration
FR	MBFIDR5	0x82c	2	Frame ID Registers
FR	MBCCFR6	0x832	2	message buffer configuration
FR	MBFIDR6	0x834	2	Frame ID Registers
FR	MBCCFR7	0x83a	2	message buffer configuration
FR	MBFIDR7	0x83c	2	Frame ID Registers
FR	MBCCFR8	0x842	2	message buffer configuration
FR	MBFIDR8	0x844	2	Frame ID Registers
FR	MBCCFR9	0x84a	2	message buffer configuration
FR	MBFIDR9	0x84c	2	Frame ID Registers
FR	MBCCFR10	0x852	2	message buffer configuration
FR	MBFIDR10	0x854	2	Frame ID Registers

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBCCFR11	0x85a	2	message buffer configuration
FR	MBFIDR11	0x85c	2	Frame ID Registers
FR	MBCCFR12	0x862	2	message buffer configuration
FR	MBFIDR12	0x864	2	Frame ID Registers
FR	MBCCFR13	0x86a	2	message buffer configuration
FR	MBFIDR13	0x86c	2	Frame ID Registers
FR	MBCCFR14	0x872	2	message buffer configuration
FR	MBFIDR14	0x874	2	Frame ID Registers
FR	MBCCFR15	0x87a	2	message buffer configuration
FR	MBFIDR15	0x87c	2	Frame ID Registers
FR	MBCCFR16	0x882	2	message buffer configuration
FR	MBFIDR16	0x884	2	Frame ID Registers
FR	MBCCFR17	0x88a	2	message buffer configuration
FR	MBFIDR17	0x88c	2	Frame ID Registers
FR	MBCCFR18	0x892	2	message buffer configuration
FR	MBFIDR18	0x894	2	Frame ID Registers
FR	MBCCFR19	0x89a	2	message buffer configuration
FR	MBFIDR19	0x89c	2	Frame ID Registers
FR	MBCCFR20	0x8a2	2	message buffer configuration
FR	MBFIDR20	0x8a4	2	Frame ID Registers
FR	MBCCFR21	0x8aa	2	message buffer configuration
FR	MBFIDR21	0x8ac	2	Frame ID Registers
FR	MBCCFR22	0x8b2	2	message buffer configuration
FR	MBFIDR22	0x8b4	2	Frame ID Registers
FR	MBCCFR23	0x8ba	2	message buffer configuration
FR	MBFIDR23	0x8bc	2	Frame ID Registers
FR	MBCCFR24	0x8c2	2	message buffer configuration
FR	MBFIDR24	0x8c4	2	Frame ID Registers
FR	MBCCFR25	0x8ca	2	message buffer configuration
FR	MBFIDR25	0x8cc	2	Frame ID Registers
FR	MBCCFR26	0x8d2	2	message buffer configuration
FR	MBFIDR26	0x8d4	2	Frame ID Registers
FR	MBCCFR27	0x8da	2	message buffer configuration
FR	MBFIDR27	0x8dc	2	Frame ID Registers
FR	MBCCFR28	0x8e2	2	message buffer configuration
FR	MBFIDR28	0x8e4	2	Frame ID Registers
FR	MBCCFR29	0x8ea	2	message buffer configuration
FR	MBFIDR29	0x8ec	2	Frame ID Registers
FR	MBCCFR30	0x8f2	2	message buffer configuration

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBFIDR30	0x8f4	2	Frame ID Registers
FR	MBCCFR31	0x8fa	2	message buffer configuration
FR	MBFIDR31	0x8fc	2	Frame ID Registers
FR	MBCCFR32	0x902	2	message buffer configuration
FR	MBFIDR32	0x904	2	Frame ID Registers
FR	MBCCFR33	0x90a	2	message buffer configuration
FR	MBFIDR33	0x90c	2	Frame ID Registers
FR	MBCCFR34	0x912	2	message buffer configuration
FR	MBFIDR34	0x914	2	Frame ID Registers
FR	MBCCFR35	0x91a	2	message buffer configuration
FR	MBFIDR35	0x91c	2	Frame ID Registers
FR	MBCCFR36	0x922	2	message buffer configuration
FR	MBFIDR36	0x924	2	Frame ID Registers
FR	MBCCFR37	0x92a	2	message buffer configuration
FR	MBFIDR37	0x92c	2	Frame ID Registers
FR	MBCCFR38	0x932	2	message buffer configuration
FR	MBFIDR38	0x934	2	Frame ID Registers
FR	MBCCFR39	0x93a	2	message buffer configuration
FR	MBFIDR39	0x93c	2	Frame ID Registers
FR	MBCCFR40	0x942	2	message buffer configuration
FR	MBFIDR40	0x944	2	Frame ID Registers
FR	MBCCFR41	0x94a	2	message buffer configuration
FR	MBFIDR41	0x94c	2	Frame ID Registers
FR	MBCCFR42	0x952	2	message buffer configuration
FR	MBFIDR42	0x954	2	Frame ID Registers
FR	MBCCFR43	0x95a	2	message buffer configuration
FR	MBFIDR43	0x95c	2	Frame ID Registers
FR	MBCCFR44	0x962	2	message buffer configuration
FR	MBFIDR44	0x964	2	Frame ID Registers
FR	MBCCFR45	0x96a	2	message buffer configuration
FR	MBFIDR45	0x96c	2	Frame ID Registers
FR	MBCCFR46	0x972	2	message buffer configuration
FR	MBFIDR46	0x974	2	Frame ID Registers
FR	MBCCFR47	0x97a	2	message buffer configuration
FR	MBFIDR47	0x97c	2	Frame ID Registers
FR	MBCCFR48	0x982	2	message buffer configuration
FR	MBFIDR48	0x984	2	Frame ID Registers
FR	MBCCFR49	0x98a	2	message buffer configuration
FR	MBFIDR49	0x98c	2	Frame ID Registers

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBCCFR50	0x992	2	message buffer configuration
FR	MBFIDR50	0x994	2	Frame ID Registers
FR	MBCCFR51	0x99a	2	message buffer configuration
FR	MBFIDR51	0x99c	2	Frame ID Registers
FR	MBCCFR52	0x9a2	2	message buffer configuration
FR	MBFIDR52	0x9a4	2	Frame ID Registers
FR	MBCCFR53	0x9aa	2	message buffer configuration
FR	MBFIDR53	0x9ac	2	Frame ID Registers
FR	MBCCFR54	0x9b2	2	message buffer configuration
FR	MBFIDR54	0x9b4	2	Frame ID Registers
FR	MBCCFR55	0x9ba	2	message buffer configuration
FR	MBFIDR55	0x9bc	2	Frame ID Registers
FR	MBCCFR56	0x9c2	2	message buffer configuration
FR	MBFIDR56	0x9c4	2	Frame ID Registers
FR	MBCCFR57	0x9ca	2	message buffer configuration
FR	MBFIDR57	0x9cc	2	Frame ID Registers
FR	MBCCFR58	0x9d2	2	message buffer configuration
FR	MBFIDR58	0x9d4	2	Frame ID Registers
FR	MBCCFR59	0x9da	2	message buffer configuration
FR	MBFIDR59	0x9dc	2	Frame ID Registers
FR	MBCCFR60	0x9e2	2	message buffer configuration
FR	MBFIDR60	0x9e4	2	Frame ID Registers
FR	MBCCFR61	0x9ea	2	message buffer configuration
FR	MBFIDR61	0x9ec	2	Frame ID Registers
FR	MBCCFR62	0x9f2	2	message buffer configuration
FR	MBFIDR62	0x9f4	2	Frame ID Registers
FR	MBCCFR63	0x9fa	2	message buffer configuration
FR	MBFIDR63	0x9fc	2	Frame ID Registers
FR	MBCCFR64	0xa02	2	message buffer configuration
FR	MBFIDR64	0xa04	2	Frame ID Registers
FR	MBCCFR65	0xa0a	2	message buffer configuration
FR	MBFIDR65	0xa0c	2	Frame ID Registers
FR	MBCCFR66	0xa12	2	message buffer configuration
FR	MBFIDR66	0xa14	2	Frame ID Registers
FR	MBCCFR67	0xa1a	2	message buffer configuration
FR	MBFIDR67	0xa1c	2	Frame ID Registers
FR	MBCCFR68	0xa22	2	message buffer configuration
FR	MBFIDR68	0xa24	2	Frame ID Registers
FR	MBCCFR69	0xa2a	2	message buffer configuration

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBFIDR69	0xa2c	2	Frame ID Registers
FR	MBCCFR70	0xa32	2	message buffer configuration
FR	MBFIDR70	0xa34	2	Frame ID Registers
FR	MBCCFR71	0xa3a	2	message buffer configuration
FR	MBFIDR71	0xa3c	2	Frame ID Registers
FR	MBCCFR72	0xa42	2	message buffer configuration
FR	MBFIDR72	0xa44	2	Frame ID Registers
FR	MBCCFR73	0xa4a	2	message buffer configuration
FR	MBFIDR73	0xa4c	2	Frame ID Registers
FR	MBCCFR74	0xa52	2	message buffer configuration
FR	MBFIDR74	0xa54	2	Frame ID Registers
FR	MBCCFR75	0xa5a	2	message buffer configuration
FR	MBFIDR75	0xa5c	2	Frame ID Registers
FR	MBCCFR76	0xa62	2	message buffer configuration
FR	MBFIDR76	0xa64	2	Frame ID Registers
FR	MBCCFR77	0xa6a	2	message buffer configuration
FR	MBFIDR77	0xa6c	2	Frame ID Registers
FR	MBCCFR78	0xa72	2	message buffer configuration
FR	MBFIDR78	0xa74	2	Frame ID Registers
FR	MBCCFR79	0xa7a	2	message buffer configuration
FR	MBFIDR79	0xa7c	2	Frame ID Registers
FR	MBCCFR80	0xa82	2	message buffer configuration
FR	MBFIDR80	0xa84	2	Frame ID Registers
FR	MBCCFR81	0xa8a	2	message buffer configuration
FR	MBFIDR81	0xa8c	2	Frame ID Registers
FR	MBCCFR82	0xa92	2	message buffer configuration
FR	MBFIDR82	0xa94	2	Frame ID Registers
FR	MBCCFR83	0xa9a	2	message buffer configuration
FR	MBFIDR83	0xa9c	2	Frame ID Registers
FR	MBCCFR84	0xaa2	2	message buffer configuration
FR	MBFIDR84	0xaa4	2	Frame ID Registers
FR	MBCCFR85	0xaaa	2	message buffer configuration
FR	MBFIDR85	0xaac	2	Frame ID Registers
FR	MBCCFR86	0xab2	2	message buffer configuration
FR	MBFIDR86	0xab4	2	Frame ID Registers
FR	MBCCFR87	0xaba	2	message buffer configuration
FR	MBFIDR87	0xabc	2	Frame ID Registers
FR	MBCCFR88	0xac2	2	message buffer configuration
FR	MBFIDR88	0xac4	2	Frame ID Registers

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBCCFR89	0xaca	2	message buffer configuration
FR	MBFIDR89	0xacc	2	Frame ID Registers
FR	MBCCFR90	0xad2	2	message buffer configuration
FR	MBFIDR90	0xad4	2	Frame ID Registers
FR	MBCCFR91	0xada	2	message buffer configuration
FR	MBFIDR91	0xadc	2	Frame ID Registers
FR	MBCCFR92	0xae2	2	message buffer configuration
FR	MBFIDR92	0xae4	2	Frame ID Registers
FR	MBCCFR93	0xaea	2	message buffer configuration
FR	MBFIDR93	0xaec	2	Frame ID Registers
FR	MBCCFR94	0xaf2	2	message buffer configuration
FR	MBFIDR94	0xaf4	2	Frame ID Registers
FR	MBCCFR95	0xafa	2	message buffer configuration
FR	MBFIDR95	0xafc	2	Frame ID Registers
FR	MBCCFR96	0xb02	2	message buffer configuration
FR	MBFIDR96	0xb04	2	Frame ID Registers
FR	MBCCFR97	0xb0a	2	message buffer configuration
FR	MBFIDR97	0xb0c	2	Frame ID Registers
FR	MBCCFR98	0xb12	2	message buffer configuration
FR	MBFIDR98	0xb14	2	Frame ID Registers
FR	MBCCFR99	0xb1a	2	message buffer configuration
FR	MBFIDR99	0xb1c	2	Frame ID Registers
FR	MBCCFR100	0xb22	2	message buffer configuration
FR	MBFIDR100	0xb24	2	Frame ID Registers
FR	MBCCFR101	0xb2a	2	message buffer configuration
FR	MBFIDR101	0xb2c	2	Frame ID Registers
FR	MBCCFR102	0xb32	2	message buffer configuration
FR	MBFIDR102	0xb34	2	Frame ID Registers
FR	MBCCFR103	0xb3a	2	message buffer configuration
FR	MBFIDR103	0xb3c	2	Frame ID Registers
FR	MBCCFR104	0xb42	2	message buffer configuration
FR	MBFIDR104	0xb44	2	Frame ID Registers
FR	MBCCFR105	0xb4a	2	message buffer configuration
FR	MBFIDR105	0xb4c	2	Frame ID Registers
FR	MBCCFR106	0xb52	2	message buffer configuration
FR	MBFIDR106	0xb54	2	Frame ID Registers
FR	MBCCFR107	0xb5a	2	message buffer configuration
FR	MBFIDR107	0xb5c	2	Frame ID Registers
FR	MBCCFR108	0xb62	2	message buffer configuration

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
FR	MBFIDR108	0xb64	2	Frame ID Registers
FR	MBCCFR109	0xb6a	2	message buffer configuration
FR	MBFIDR109	0xb6c	2	Frame ID Registers
FR	MBCCFR110	0xb72	2	message buffer configuration
FR	MBFIDR110	0xb74	2	Frame ID Registers
FR	MBCCFR111	0xb7a	2	message buffer configuration
FR	MBFIDR111	0xb7c	2	Frame ID Registers
FR	MBCCFR112	0xb82	2	message buffer configuration
FR	MBFIDR112	0xb84	2	Frame ID Registers
FR	MBCCFR113	0xb8a	2	message buffer configuration
FR	MBFIDR113	0xb8c	2	Frame ID Registers
FR	MBCCFR114	0xb92	2	message buffer configuration
FR	MBFIDR114	0xb94	2	Frame ID Registers
FR	MBCCFR115	0xb9a	2	message buffer configuration
FR	MBFIDR115	0xb9c	2	Frame ID Registers
FR	MBCCFR116	0xba2	2	message buffer configuration
FR	MBFIDR116	0xba4	2	Frame ID Registers
FR	MBCCFR117	0xbaa	2	message buffer configuration
FR	MBFIDR117	0xbac	2	Frame ID Registers
FR	MBCCFR118	0xbb2	2	message buffer configuration
FR	MBFIDR118	0xbb4	2	Frame ID Registers
FR	MBCCFR119	0xbba	2	message buffer configuration
FR	MBFIDR119	0xbbc	2	Frame ID Registers
FR	MBCCFR120	0xbc2	2	message buffer configuration
FR	MBFIDR120	0xbc4	2	Frame ID Registers
FR	MBCCFR121	0xbca	2	message buffer configuration
FR	MBFIDR121	0xbcc	2	Frame ID Registers
FR	MBCCFR122	0xbd2	2	message buffer configuration
FR	MBFIDR122	0xbd4	2	Frame ID Registers
FR	MBCCFR123	0xbda	2	message buffer configuration
FR	MBFIDR123	0xbdc	2	Frame ID Registers
FR	MBCCFR124	0xbe2	2	message buffer configuration
FR	MBFIDR124	0xbe4	2	Frame ID Registers
FR	MBCCFR125	0xbea	2	message buffer configuration
FR	MBFIDR125	0xbec	2	Frame ID Registers
FR	MBCCFR126	0xbf2	2	message buffer configuration
FR	MBFIDR126	0xbf4	2	Frame ID Registers
FR	MBCCFR127	0xbfa	2	message buffer configuration
FR	MBFIDR127	0xbfc	2	Frame ID Registers

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
I2C_1	IBAD	0x00	1	Bus Address Register
I2C_1	IBFD	0x01	1	Bus Frequency Divider Register
I2C_1	IBCR	0x02	1	Bus Control Register
I2C_1	IBIC	0x05	1	Bus Interrupt Config Register
I2C_2	IBAD	0x00	1	Bus Address Register
I2C_2	IBFD	0x01	1	Bus Frequency Divider Register
I2C_2	IBCR	0x02	1	Bus Control Register
I2C_2	IBIC	0x05	1	Bus Interrupt Config Register
JTAGM	MCR	0x00	3	Module configuration register
LINFlexD_1	LINCR1	0x00	3	LIN Control Register
LINFlexD_1	LINIER	0x04	3	LIN Interrupt enable Register
LINFlexD_1	UARTCR	0x10	3	UART Mode Control Register
LINFlexD_1	LINTCSR	0x18	3	LIN Time-Out Control status register (it contains 3 flags used to configure the time out)
LINFlexD_1	LINOCR	0x1c	3	LIN Output Compare Register; compare values configuration
LINFlexD_1	LINTOCR	0x20	3	LIN Time-Out Control Register; timeout configuration values
LINFlexD_1	LINFBRR	0x24	3	baud rate configuration
LINFlexD_1	LINIBRR	0x28	3	baud rate configuration
LINFlexD_1	LINCR2	0x30	3	configures the LINFlexD_1 state machine behaviour in case of error
LINFlexD_1	BIDR	0x34	3	buffer identifier register
LINFlexD_1	IFER	0x40	3	identifier filter enable register
LINFlexD_1	IFMR	0x48	3	identifier filter mode register
LINFlexD_1	IFCR0	0x4C	3	Identifier Filter Control Register
LINFlexD_1	IFCR1	0x50	3	Identifier Filter Control Register
LINFlexD_1	IFCR2	0x54	3	Identifier Filter Control Register
LINFlexD_1	IFCR3	0x58	3	Identifier Filter Control Register
LINFlexD_1	IFCR4	0x5C	3	Identifier Filter Control Register
LINFlexD_1	IFCR5	0x60	3	Identifier Filter Control Register
LINFlexD_1	IFCR6	0x64	3	Identifier Filter Control Register
LINFlexD_1	IFCR7	0x68	3	Identifier Filter Control Register
LINFlexD_1	IFCR8	0x6C	3	Identifier Filter Control Register
LINFlexD_1	IFCR9	0x70	3	Identifier Filter Control Register
LINFlexD_1	IFCR10	0x74	3	Identifier Filter Control Register
LINFlexD_1	IFCR11	0x78	3	Identifier Filter Control Register
LINFlexD_1	IFCR12	0x7C	3	Identifier Filter Control Register
LINFlexD_1	IFCR13	0x80	3	Identifier Filter Control Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
LINFlexD_1	IFCR14	0x84	3	Identifier Filter Control Register
LINFlexD_1	IFCR15	0x88	3	Identifier Filter Control Register
LINFlexD_1	GCR	0x8c	3	Global Control Register
LINFlexD_1	UARTPTO	0x90	3	UART Preset Time Out Register
LINFlexD_1	DMATXE	0x98	3	DMA Tx Enable Register
LINFlexD_1	DMARXE	0x9c	3	DMA Rx Enable Register
CMU_0	CSR	0x200	3	Control Status Register
CMU_0	HFREFR	0x208	3	High Frequency Reference Register
CMU_0	LFREFR	0x20C	3	Low Frequency Reference Register
CMU_0	MDR	0x218	3	Measurement Duration Register
CMU_1	CSR	0x240	3	Control Status Register
CMU_1	HFREFR	0x248	3	High Frequency Reference Register
CMU_1	LFREFR	0x24C	3	Low Frequency Reference Register
CMU_1	MDR	0x258	3	Measurement Duration Register
CMU_2	CSR	0x280	3	Control Status Register
CMU_2	HFREFR	0x288	3	High Frequency Reference Register
CMU_2	LFREFR	0x28C	3	Low Frequency Reference Register
CMU_2	MDR	0x298	3	Measurement Duration Register
CMU_3	CSR	0x2C0	3	Control Status Register
CMU_3	HFREFR	0x2C8	3	High Frequency Reference Register
CMU_3	LFREFR	0x2CC	3	Low Frequency Reference Register
CMU_3	MDR	0x2D8	3	Measurement Duration Register
CMU_4	CSR	300	3	Control Status Register
CMU_4	HFREFR	308	3	High Frequency Reference Register
CMU_4	LFREFR	30C	3	Low Frequency Reference Register
CMU_4	MDR	318	3	Measurement Duration Register
CMU_5	CSR	0x340	3	Control Status Register
CMU_5	HFREFR	0x348	3	High Frequency Reference Register
CMU_5	LFREFR	0x34C	3	Low Frequency Reference Register
CMU_5	MDR	0x358	3	Measurement Duration Register
CMU_6	CSR	0x380	3	Control Status Register
CMU_6	HFREFR	0x388	3	High Frequency Reference Register
CMU_6	LFREFR	0x38C	3	Low Frequency Reference Register
CMU_6	MDR	0x398	3	Measurement Duration Register
CMU_7	CSR	0x3C0	3	Control Status Register
CMU_7	HFREFR	0x3C8	3	High Frequency Reference Register
CMU_7	LFREFR	0x3CC	3	Low Frequency Reference Register
CMU_7	MDR	0x3D8	3	Measurement Duration Register
CMU_8	CSR	0x400	3	Control Status Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CMU_8	HFREFR	0x408	3	High Frequency Reference Register
CMU_8	LFREFR	0x40C	3	Low Frequency Reference Register
CMU_8	MDR	0x418	3	Measurement Duration Register
CMU_9	CSR	0x440	3	Control Status Register
CMU_9	HFREFR	0x448	3	High Frequency Reference Register
CMU_9	LFREFR	0x44C	3	Low Frequency Reference Register
CMU_9	MDR	0x458	3	Measurement Duration Register
CMU_10	CSR	0x480	3	Control Status Register
CMU_10	HFREFR	0x488	3	High Frequency Reference Register
CMU_10	LFREFR	0x48C	3	Low Frequency Reference Register
CMU_10	MDR	0x498	3	Measurement Duration Register
CMU_11	CSR	0x4C0	3	Control Status Register
CMU_11	HFREFR	0x4C8	3	High Frequency Reference Register
CMU_11	LFREFR	0x4CC	3	Low Frequency Reference Register
CMU_11	MDR	0x4D8	3	Measurement Duration Register
CMU_12	CSR	0x500	3	Control Status Register
CMU_12	HFREFR	0x508	3	High Frequency Reference Register
CMU_12	LFREFR	0x50C	3	Low Frequency Reference Register
CMU_12	MDR	0x518	3	Measurement Duration Register
CGM	PCS_SDUR	0x700	1	PCS Switch Duration Register
CGM	PCS_DIVC1	0x704	3	PCS Divider Change Register 1
CGM	PCS_DIVE1	0x708	3	PCS Divider End Register 1
CGM	PCS_DIVS1	0x70C	3	PCS Divider Start Register 1
CGM	PCS_DIVC2	0x710	3	PCS Divider Change Register 2
CGM	PCS_DIVE2	0x714	3	PCS Divider End Register 2
CGM	PCS_DIVS2	0x718	3	PCS Divider Start Register 2
CGM	PCS_DIVC4	0x728	3	PCS Divider Change Register 4
CGM	PCS_DIVE4	0x72C	3	PCS Divider End Register 4
CGM	PCS_DIVS4	0x730	3	PCS Divider Start Register 4
CGM	SC_DC0	0x7E8	3	System Clock Divider 0 Configuration Register
CGM	SC_DC1	0x7EC	3	System Clock Divider 1 Configuration Register
CGM	SC_DC2	0x7F0	3	System Clock Divider 2 Configuration Register
CGM	SC_DC3	0x7F4	3	System Clock Divider 3 Configuration Register
CGM	SC_DC4	0x7F8	3	System Clock Divider 4 Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
CGM	SC_DC5	0x7FC	3	System Clock Divider 5 Configuration Register
CGM	AC0_SC	0x800	3	Auxiliary Clock Select Control Register
CGM	AC0_DC0	0x808	3	Auxiliary Clock 0 Divider 0 Configuration Register
CGM	AC0_DC2	0x810	3	Auxiliary Clock 0 Divider 2 Configuration Register
CGM	AC1_SC	0x820	3	Auxiliary Clock 1 Select Control Register
CGM	AC1_DC0	0x828	3	Auxiliary Clock 1 Divider 0 Configuration Register
CGM	AC2_SC	0x840	3	Auxiliary Clock 2 Select Control Register
CGM	AC2_DC0	0x848	3	Auxiliary Clock 2 Divider 0 Configuration Register
CGM	AC3_SC	0x860	3	Auxiliary Clock 3 Select Control Register
CGM	AC4_SC	0x880	3	Auxiliary Clock 4 Select Control Register
CGM	AC7_SC	0x8E0	3	Auxiliary Clock 7 Select Control Register
CGM	AC7_DC0	0x8E8	3	Auxiliary Clock 7 Divider 0 Configuration Register
CGM	AC8_SC	0x900	3	Auxiliary Clock 8 Select Control Register
CGM	AC8_DC0	0x908	3	Auxiliary Clock 8 Divider 0 Configuration Register
CGM	AC9_DC0	0x928	3	Auxiliary Clock 9 Divider 0 Configuration Register
CGM	AC10_SC	0x940	3	Auxiliary Clock 10 Select Control Register
CGM	AC10_DC0	0x948	3	Auxiliary Clock 10 Divider 0 Configuration Register
CGM	AC11_SC	0x960	3	Auxiliary Clock 11 Select Control Register
CGM	AC11_DC0	0x968	3	Auxiliary Clock 11 Divider 0 Configuration Register
CGM	AC12_SC	0x980	3	Auxiliary Clock 12 Select Control Register
CGM	AC12_DC0	0x988	3	Auxiliary Clock 12 Divider 0 Configuration Register
CGM	AC13_SC	0x9A0	3	Auxiliary Clock 13 Select Control Register
CGM	AC13_DC0	0x9A8	3	Auxiliary Clock 13 Divider 0 Configuration Register
CGM	AC14_DC0	0x9C8	3	Auxiliary Clock 14 Divider 0 Configuration Register
IRCOSC	CTL	0x00	3	Control register
MC_ME	ME	0x08	3	Mode Enable Register
MC_ME	IM	0x10	3	Interrupt Mask Register
MC_ME	TEST_MC	0x24	3	TEST Mode Configuration Register
MC_ME	SAFE_MC	0x28	3	SAFE Mode Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
MC_ME	DRUN_MC	0x2C	3	DRUN Mode Configuration Register
MC_ME	RUN0_MC	0x30	3	RUN0 3 Mode Configuration Register
MC_ME	RUN1_MC	0x34	3	RUN0 3 Mode Configuration Register
MC_ME	RUN2_MC	0x38	3	RUN0 3 Mode Configuration Register
MC_ME	RUN3_MC	0x3C	3	RUN0 3 Mode Configuration Register
MC_ME	HALT0_MC	0x40	3	HALT0 Mode Configuration Register
MC_ME	STOP0_MC	0x48	3	STOP0 Mode Configuration Register
MC_ME	RUN_PC0	0x80	3	Run Peripheral Configuration Register
MC_ME	RUN_PC1	0x84	3	Run Peripheral Configuration Register
MC_ME	RUN_PC2	0x88	3	Run Peripheral Configuration Register
MC_ME	RUN_PC3	0x8C	3	Run Peripheral Configuration Register
MC_ME	RUN_PC4	0x90	3	Run Peripheral Configuration Register
MC_ME	RUN_PC5	0x94	3	Run Peripheral Configuration Register
MC_ME	RUN_PC6	0x98	3	Run Peripheral Configuration Register
MC_ME	RUN_PC7	0x9C	3	Run Peripheral Configuration Register
MC_ME	LP_PC0	0xA0	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC1	0xA4	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC2	0xA8	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC3	0xAC	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC4	0xB0	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC5	0xB4	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC6	0xB8	3	Low-Power Peripheral Configuration Register
MC_ME	LP_PC7	0xBC	3	Low-Power Peripheral Configuration Register
MC_ME	PCTL9	0xC9	1	LFAST_0
MC_ME	PCTL11	0xCB	1	SIPI_0
MC_ME	PCTL12	0xCC	1	ENET_0
MC_ME	PCTL30	0xDE	1	PIT_0
MC_ME	PCTL31	0xDF	1	PIT_1
MC_ME	PCTL36	0xE4	1	DMAMUX_0
MC_ME	PCTL38	0xE6	1	CRC_0
MC_ME	PCTL49	0xF1	1	DTS
MC_ME	PCTL61	0xFD	1	MIPI
MC_ME	PCTL77	0x10D	1	CAN_2
MC_ME	PCTL78	0x10E	1	CAN_1
MC_ME	PCTL79	0x10F	1	CAN_0
MC_ME	PCTL91	0x11B	1	LINFlexD_1_1
MC_ME	PCTL98	0x122	1	SPI_1
MC_ME	PCTL100	0x124	1	I2C_2
MC_ME	PCTL102	0x126	1	I2C_1

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
MC_ME	PCTL107	0x12B	1	FR
MC_ME	PCTL126	0x13E	1	SARADC_1
MC_ME	PCTL137	0x149	1	ETIMER_1
MC_ME	PCTL146	0x152	1	DMAMUX_1
MC_ME	PCTL148	0x154	1	CRC_1
MC_ME	PCTL187	0x17B	1	SPT
MC_ME	PCTL188	0x17C	1	CTE
MC_ME	PCTL209	0x191	1	SPI_2
MC_ME	PCTL237	0x1AD	1	SARADC_0
MC_ME	PCTL238	0x1AE	1	WGM
MC_ME	PCTL245	0x1B5	1	ETIMER_2
MC_ME	PCTL251	0x1BB	1	CTU_0
MC_ME	PCTL255	0x1BF	1	FlexPWM_0
RGM	DERD	0x10	3	‘Destructive’ Event Reset Disable Register
RGM	DEAR	0x20	3	‘Destructive’ Event Alternate Request Register
RGM	FERD	0x310	3	‘Functional’ Event Reset Disable Register
RGM	FEAR	0x320	3	‘Functional’ Event Alternate Request Register
RGM	FBRE	0x330	3	‘Functional’ Bidirectional Reset Enable Register
RGM	FESS	0x340	3	‘Functional’ Event Short Sequence Register
RGM	FRET	0x604	1	'Functional’ Event Short Sequence Register
memu	CTRL	0x00	3	Control register
memu	DEBUG	0x0C	3	Debug register
PIT_0	MCR	0x00	3	Module Control Register
PIT_0	LDVAL0	0x100	3	Load Value Register
PIT_0	TCTRL0	0x108	3	Timer Control Register
PIT_0	LDVAL1	0x110	3	Load Value Register
PIT_0	TCTRL1	0x118	3	Timer Control Register
PIT_0	LDVAL2	0x120	3	Load Value Register
PIT_0	TCTRL2	0x128	3	Timer Control Register
PIT_0	LDVAL3	0x130	3	Load Value Register
PIT_0	TCTRL3	0x138	3	Timer Control Register
PIT_1	MCR	0x00	3	Module Control Register
PIT_1	LDVAL0	0x100	3	Load Value Register
PIT_1	TCTRL0	0x108	3	Timer Control Register
PIT_1	LDVAL1	0x110	3	Load Value Register
PIT_1	TCTRL1	0x118	3	Timer Control Register
PIT_1	LDVAL2	0x120	3	Load Value Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
PIT_1	TCTRL2	0x128	3	Timer Control Register
PIT_1	LDVAL3	0x130	3	Load Value Register
PIT_1	TCTRL3	0x138	3	Timer Control Register
PLLDIG	PLL0CR	0x00	3	PLL0 Control Register
PLLDIG	PLL0DV	0x08	3	PLL0 Divider Register
PLLDIG	PLL1CR	0x20	3	PLL1 Control Register
PLLDIG	PLL1DV	0x28	3	PLL1 Divider Register
PLLDIG	PLL1FM	0x2C	3	modulation configuration
PLLDIG	PLL1FD	0x30	3	fractional divide control
PMC	SMPS_CNTRL	0x64	3	PMC Control Register
PMC	STTW	0x68	3	LVD Self Test Time Window Register
PMC	SELF_TEST_UM_VD_REG	0x6C	3	Voltage Detect User Mode Test Register
PMC	AFE_INTR_ENA	0x78	3	Interrupt Enable Register
PMC	FIR	0x8C	3	FCCU Fault Injection Register
PMC	PMCCR	0x94	3	PMC Control Register
PMC	TS_IER	0x98	2	Interrupt Enable Register
PMC	REE_TD	0xA0	3	Temperature Reset Event Enable
PMC	RES_TD	0xA4	3	Temperature Reset Event Selection register
PMC	CTL_TD	0xA8	3	Temperature detector configuration register
PMC	TS_FIR	0xB4	2	Fault Injection Register
SIPI	CCR0	0x00	3	SIPI Channel Control Register 0
SIPI	CIR0	0x0C	3	SIPI Channel Interrupt Register 0
SIPI	CTOR0	0x10	3	SIPI Channel Timeout Register 0
SIPI	CCR1	0x20	3	SIPI Channel Control Register 1
SIPI	CIR1	0x2C	3	SIPI Channel Interrupt Register 1
SIPI	CTOR1	0x30	3	SIPI Channel Timeout Register 1
SIPI	CCR2	0x40	3	SIPI Channel Control Register 2
SIPI	CIR2	0x4C	3	SIPI Channel Interrupt Register 2
SIPI	CTOR2	0x50	3	SIPI Channel Timeout Register 2
SIPI	CCR3	0x7C	3	SIPI Channel Control Register 3
SIPI	CIR3	0x88	3	SIPI Channel Interrupt Register 3
SIPI	CTOR3	0x8C	3	SIPI Channel Timeout Register 3
SIPI	MCR	0x9C	3	SIPI Module Configuration Register
SIUL2	DIRER0	0x18	3	DMA/Interrupt Request Enable Register0
SIUL2	DIRSR0	0x20	3	DMA/Interrupt Request Select Register0
SIUL2	IREER0	0x28	3	Interrupt Rising-Edge Event Enable Register 0
SIUL2	IFEER0	0x30	3	Interrupt Falling-Edge Event Enable Register 0

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SIUL2	IFER0	0x38	3	Interrupt Filter Enable Register 0
SIUL2	IFMCR0	0x40	3	Interrupt Filter Maximum Counter 0
SIUL2	IFMCR1	0x44	3	Interrupt Filter Maximum Counter 1
SIUL2	IFMCR2	0x48	3	Interrupt Filter Maximum Counter 2
SIUL2	IFMCR3	0x4c	3	Interrupt Filter Maximum Counter 3
SIUL2	IFMCR4	0x50	3	Interrupt Filter Maximum Counter 4
SIUL2	IFMCR5	0x54	3	Interrupt Filter Maximum Counter 5
SIUL2	IFMCR6	0x58	3	Interrupt Filter Maximum Counter 6
SIUL2	IFMCR7	0x5c	3	Interrupt Filter Maximum Counter 7
SIUL2	IFMCR8	0x60	3	Interrupt Filter Maximum Counter 8
SIUL2	IFMCR9	0x64	3	Interrupt Filter Maximum Counter 9
SIUL2	IFMCR10	0x68	3	Interrupt Filter Maximum Counter 10
SIUL2	IFMCR11	0x6c	3	Interrupt Filter Maximum Counter 11
SIUL2	IFMCR12	0x70	3	Interrupt Filter Maximum Counter 12
SIUL2	IFMCR13	0x74	3	Interrupt Filter Maximum Counter 13
SIUL2	IFMCR14	0x78	3	Interrupt Filter Maximum Counter 14
SIUL2	IFMCR15	0x7c	3	Interrupt Filter Maximum Counter 15
SIUL2	IFMCR16	0x80	3	Interrupt Filter Maximum Counter 16
SIUL2	IFMCR17	0x84	3	Interrupt Filter Maximum Counter 17
SIUL2	IFMCR18	0x88	3	Interrupt Filter Maximum Counter 18
SIUL2	IFMCR19	0x8c	3	Interrupt Filter Maximum Counter 19
SIUL2	IFMCR20	0x90	3	Interrupt Filter Maximum Counter 20
SIUL2	IFMCR21	0x94	3	Interrupt Filter Maximum Counter 21
SIUL2	IFMCR22	0x98	3	Interrupt Filter Maximum Counter 22
SIUL2	IFMCR23	0x9c	3	Interrupt Filter Maximum Counter 23
SIUL2	IFMCR24	0xa0	3	Interrupt Filter Maximum Counter 24
SIUL2	IFMCR25	0xa4	3	Interrupt Filter Maximum Counter 25
SIUL2	IFMCR26	0xa8	3	Interrupt Filter Maximum Counter 26
SIUL2	IFMCR27	0xac	3	Interrupt Filter Maximum Counter 27
SIUL2	IFMCR28	0xb0	3	Interrupt Filter Maximum Counter 28
SIUL2	IFMCR29	0xb4	3	Interrupt Filter Maximum Counter 29
SIUL2	IFMCR30	0xb8	3	Interrupt Filter Maximum Counter 30
SIUL2	IFMCR31	0xbc	3	Interrupt Filter Maximum Counter 31
SIUL2	IFCPR	0xc0	3	Interrupt Filter Clock Prescaler
SIUL2	MSCR0	0x240	3	Multiplexed Signal Configuration Register
SIUL2	MSCR1	0x244	3	Multiplexed Signal Configuration Register
SIUL2	MSCR2	0x248	3	Multiplexed Signal Configuration Register
SIUL2	MSCR3	0x24C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR4	0x250	3	Multiplexed Signal Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SIUL2	MSCR5	0x254	3	Multiplexed Signal Configuration Register
SIUL2	MSCR6	0x258	3	Multiplexed Signal Configuration Register
SIUL2	MSCR7	0x25C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR8	0x260	3	Multiplexed Signal Configuration Register
SIUL2	MSCR9	0x264	3	Multiplexed Signal Configuration Register
SIUL2	MSCR10	0x268	3	Multiplexed Signal Configuration Register
SIUL2	MSCR11	0x26C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR12	0x270	3	Multiplexed Signal Configuration Register
SIUL2	MSCR13	0x274	3	Multiplexed Signal Configuration Register
SIUL2	MSCR14	0x278	3	Multiplexed Signal Configuration Register
SIUL2	MSCR15	0x27C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR16	0x280	3	Multiplexed Signal Configuration Register
SIUL2	MSCR17	0x284	3	Multiplexed Signal Configuration Register
SIUL2	MSCR18	0x288	3	Multiplexed Signal Configuration Register
SIUL2	MSCR19	0x28C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR20	0x290	3	Multiplexed Signal Configuration Register
SIUL2	MSCR21	0x294	3	Multiplexed Signal Configuration Register
SIUL2	MSCR22	0x298	3	Multiplexed Signal Configuration Register
SIUL2	MSCR23	0x29C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR24	0x2A0	3	Multiplexed Signal Configuration Register
SIUL2	MSCR25	0x2A4	3	Multiplexed Signal Configuration Register
SIUL2	MSCR26	0x2A8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR27	0x2AC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR28	0x2B0	3	Multiplexed Signal Configuration Register
SIUL2	MSCR29	0x2B4	3	Multiplexed Signal Configuration Register
SIUL2	MSCR30	0x2B8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR31	0x2BC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR32	0x2C0	3	Multiplexed Signal Configuration Register
SIUL2	MSCR33	0x2C4	3	Multiplexed Signal Configuration Register
SIUL2	MSCR34	0x2C8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR42	0x2E8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR43	0x2EC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR44	0x2F0	3	Multiplexed Signal Configuration Register
SIUL2	MSCR47	0x2FC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR48	0x300	3	Multiplexed Signal Configuration Register
SIUL2	MSCR49	0x304	3	Multiplexed Signal Configuration Register
SIUL2	MSCR50	0x308	3	Multiplexed Signal Configuration Register
SIUL2	MSCR51	0x30C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR52	0x310	3	Multiplexed Signal Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SIUL2	MSCR53	0x314	3	Multiplexed Signal Configuration Register
SIUL2	MSCR54	0x318	3	Multiplexed Signal Configuration Register
SIUL2	MSCR56	0x320	3	Multiplexed Signal Configuration Register
SIUL2	MSCR59	0x32C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR62	0x338	3	Multiplexed Signal Configuration Register
SIUL2	MSCR66	0x348	3	Multiplexed Signal Configuration Register
SIUL2	MSCR68	0x350	3	Multiplexed Signal Configuration Register
SIUL2	MSCR69	0x354	3	Multiplexed Signal Configuration Register
SIUL2	MSCR70	0x358	3	Multiplexed Signal Configuration Register
SIUL2	MSCR71	0x35C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR77	0x374	3	Multiplexed Signal Configuration Register
SIUL2	MSCR79	0x37C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR80	0x380	3	Multiplexed Signal Configuration Register
SIUL2	MSCR94	0x3B8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR95	0x3BC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR101	0x3D4	3	Multiplexed Signal Configuration Register
SIUL2	MSCR104	0x3E0	3	Multiplexed Signal Configuration Register
SIUL2	MSCR105	0x3E4	3	Multiplexed Signal Configuration Register
SIUL2	MSCR106	0x3E8	3	Multiplexed Signal Configuration Register
SIUL2	MSCR107	0x3EC	3	Multiplexed Signal Configuration Register
SIUL2	MSCR116	0x410	3	Multiplexed Signal Configuration Register
SIUL2	MSCR117	0x414	3	Multiplexed Signal Configuration Register
SIUL2	MSCR118	0x418	3	Multiplexed Signal Configuration Register
SIUL2	MSCR119	0x41C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR120	0x420	3	Multiplexed Signal Configuration Register
SIUL2	MSCR121	0x424	3	Multiplexed Signal Configuration Register
SIUL2	MSCR123	0x42C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR124	0x430	3	Multiplexed Signal Configuration Register
SIUL2	MSCR125	0x434	3	Multiplexed Signal Configuration Register
SIUL2	MSCR128	0x440	3	Multiplexed Signal Configuration Register
SIUL2	MSCR129	0x444	3	Multiplexed Signal Configuration Register
SIUL2	MSCR130	0x448	3	Multiplexed Signal Configuration Register
SIUL2	MSCR132	0x450	3	Multiplexed Signal Configuration Register
SIUL2	MSCR133	0x454	3	Multiplexed Signal Configuration Register
SIUL2	MSCR134	0x458	3	Multiplexed Signal Configuration Register
SIUL2	MSCR135	0x45C	3	Multiplexed Signal Configuration Register
SIUL2	MSCR136	0x460	3	Multiplexed Signal Configuration Register
SIUL2	MSCR137	0x464	3	Multiplexed Signal Configuration Register
SIUL2	MSCR198	0x558	3	Multiplexed Signal Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SIUL2	IMCR0	0xA40	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR1	0xA44	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR2	0xA48	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR4	0xA50	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR5	0xA54	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR6	0xA58	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR7	0xA5c	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR8	0xA60	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR9	0xA64	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR10	0xA68	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR20	0xA90	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR27	0xAAc	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR28	0xAB0	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR29	0xAB4	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR30	0xAB8	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR31	0xABc	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR32	0xAC0	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR33	0xAC4	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR34	0xAC8	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR35	0xACc	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR36	0xAD0	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR37	0xAD4	3	Input Multiplexed Signal Configuration Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SIUL2	IMCR38	0xAD8	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR39	0xADc	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR40	0xAE0	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR41	0xAE4	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR42	0xAE8	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR43	0xAEc	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR52	0xB10	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR53	0xB14	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR54	0xB18	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR60	0xB30	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR61	0xB34	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR63	0xB3c	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR88	0xBA0	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR89	0xBA4	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR90	0xBA8	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR91	0xBAC	3	Input Multiplexed Signal Configuration Register
SIUL2	IMCR93	0xBB4	3	Input Multiplexed Signal Configuration Register
SPT	SPT_GBL_CTRL	0x00	3	SPT Global Control Register
SPT	ACQ_GBL_CTRL_0	0x04	3	Acquistion Global Control Register 0
SPT	ACQ_GBL_CTRL_1	0x08	3	Acquistion Global Control Register 1
SPT	SDMA_CTRL0	0x20	3	SDMA Control 0 Register
SPT	SDMA_CTRL1	0x24	3	SDMA Control 1 Register
SPT	ACQ_CSI_CTRL	0xA4	3	Acquistion mipi-csi control register
SPT	SDMA_BYP_CTRL0	0xA8	3	Bypass SDMA Control 0
SPT	SDMA_BYP_CTRL1	0xAC	3	Bypass SDMA Control 1
SPT	ACQ_BYP_CTRL	0xB0	3	Acquistion Bypass Control Register
SPT	CS_PG_ST_ADDR	0xC0	3	Program Start Addr Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
SPT	CS_MODE_CTRL	0xC4	3	Mode Control Register
SPT	PDMA_LFSR_LOAD_VAL_HIGH	0x180	3	LFSR Load High Value
SPT	PDMA_LFSR_LOAD_VAL_LOW	0x184	3	LFSR Load Low Value
SPT	PDMA_CONTROL	0x188	3	PDMA Control Register
SPT	MEM_ERR_INJECT_CTRL	0x1FC	3	Memory Error Injection Register
SPT	MEM_ERR_INT_EN	0x204	3	Memory Interrupt Enable register
SPT	DMA_ERR_INT_EN	0x20C	3	DMA_ERROR_STATUS Interrupt Enable
SPT	GBL_STATUS_IE	0x214	3	Global Status Interrupt Enable Register
SPT	HW_ACC_ERR_IE	0x21C	3	Hardware Accelerator Error Interrupt Enable Register
SPT	HIST_OVF_IE	0x228	3	HIST Overflow Interrupt Enable Register
SPT	CS_INTEN0	0x22C	3	Interrupt Enable Register 0
SPT	CS_INTEN1	0x230	3	Interrupt Enable Register 1
SPT	CS_EVT1_INTEN	0x234	3	EVT1 Interrupt Enable Register
SPT	CS_EVT2_INTEN	0x238	3	EVT2 Interrupt Enable Register
SPT	WR_0_15_CTRL_REG	0x240	3	Work register
SPT	WR_16_31_CTRL_REG	0x244	3	Work register
SPT	WR_32_47_CTRL_REG	0x248	3	Work register
SSCM	ERROR	0x06	2	error configuration register
STCU	RUN	0x00	3	Run Register
STCU	RUNSW	0x04	3	Run Software Register
STCU	CFG	0x0C	3	Configuration Register
STCU	PLL_CFG	0x10	3	PLL Configuration Register
STCU	WDG	0x14	3	Watchdog Register Granularity
STCU	LBRMSW	0x3c	3	On-Line LBIST Reset Management
STCU	LBUFM	0x40	3	LBIST Unrecoverable FM Register
STCU	MBUFML	0x74	3	MBIST Unrecoverable FM Low Register
STCU	LB_CTRL0	0x100	3	LBIST Control Register
STCU	LB_PCS0	0x104	3	LBIST PC Stop Register
STCU	LB_CTRL1	0x140	3	LBIST Control Register
STCU	LB_PCS1	0x144	3	LBIST PC Stop Register
STCU	LB_CTRL2	0x180	3	LBIST Control Register
STCU	LB_PCS2	0x184	3	LBIST PC Stop Register
STCU	LB_CTRL3	0x1c0	3	LBIST Control Register
STCU	LB_PCS3	0x1c4	3	LBIST PC Stop Register
STCU	LB_CTRL4	0x200	3	LBIST Control Register
STCU	LB_PCS4	0x204	3	LBIST PC Stop Register
STCU	LB_CTRL5	0x240	3	LBIST Control Register
STCU	LB_PCS5	0x244	3	LBIST PC Stop Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
STCU	LB_CTRL6	0x280	3	LBIST Control Register
STCU	LB_PCS6	0x284	3	LBIST PC Stop Register
STCU	LB_CTRL7	0x2c0	3	LBIST Control Register
STCU	LB_PCS7	0x2c4	3	LBIST PC Stop Register
STCU	MB_CTRL0	0x600	3	MBIST Control Register
STCU	MB_CTRL1	0x604	3	MBIST Control Register
STCU	MB_CTRL2	0x608	3	MBIST Control Register
STCU	MB_CTRL3	0x60C	3	MBIST Control Register
STCU	MB_CTRL4	0x610	3	MBIST Control Register
STCU	MB_CTRL5	0x614	3	MBIST Control Register
STCU	MB_CTRL6	0x618	3	MBIST Control Register
STCU	MB_CTRL7	0x61C	3	MBIST Control Register
STCU	MB_CTRL8	0x620	3	MBIST Control Register
STCU	MB_CTRL9	0x624	3	MBIST Control Register
STCU	MB_CTRL10	0x628	3	MBIST Control Register
STCU	MB_CTRL11	0x62C	3	MBIST Control Register
STCU	MB_CTRL12	0x630	3	MBIST Control Register
STCU	MB_CTRL13	0x634	3	MBIST Control Register
STCU	MB_CTRL14	0x638	3	MBIST Control Register
STCU	MB_CTRL15	0x63C	3	MBIST Control Register
STCU	MB_CTRL16	0x640	3	MBIST Control Register
STCU	MB_CTRL17	0x644	3	MBIST Control Register
STCU	MB_CTRL18	0x648	3	MBIST Control Register
STCU	MB_CTRL19	0x64C	3	MBIST Control Register
STCU	MB_CTRL20	0x650	3	MBIST Control Register
STCU	MB_CTRL21	0x654	3	MBIST Control Register
STCU	MB_CTRL22	0x658	3	MBIST Control Register
STCU	MB_CTRL23	0x65C	3	MBIST Control Register
STCU	MB_CTRL24	0x660	3	MBIST Control Register
STCU	MB_CTRL25	0x664	3	MBIST Control Register
STCU	MB_CTRL26	0x668	3	MBIST Control Register
STCU	MB_CTRL27	0x66C	3	MBIST Control Register
STCU	MB_CTRL28	0x670	3	MBIST Control Register
STCU	MB_CTRL29	0x674	3	MBIST Control Register
STCU	MB_CTRL30	0x678	3	MBIST Control Register
STCU	MB_CTRL31	0x67C	3	MBIST Control Register
STCU	MB_CTRL32	0x680	3	MBIST Control Register
STCU	MB_CTRL33	0x684	3	MBIST Control Register
STCU	MB_CTRL34	0x688	3	MBIST Control Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
STCU	MB_CTRL35	0x68C	3	MBIST Control Register
STCU	MB_CTRL36	0x690	3	MBIST Control Register
STCU	MB_CTRL37	0x694	3	MBIST Control Register
STCU	MB_CTRL38	0x698	3	MBIST Control Register
STCU	MB_CTRL39	0x69C	3	MBIST Control Register
STCU	MB_CTRL40	0x6A0	3	MBIST Control Register
STCU	MB_CTRL41	0x6A4	3	MBIST Control Register
STCU	MB_CTRL42	0x6A8	3	MBIST Control Register
STCU	MB_CTRL43	0x6AC	3	MBIST Control Register
STCU	MB_CTRL44	0x6B0	3	MBIST Control Register
STCU	MB_CTRL45	0x6B4	3	MBIST Control Register
STCU	MB_CTRL46	0x6B8	3	MBIST Control Register
STCU	MB_CTRL47	0x6BC	3	MBIST Control Register
STCU	MB_CTRL48	0x6C0	3	MBIST Control Register
STCU	MB_CTRL49	0x6C4	3	MBIST Control Register
STCU	MB_CTRL50	0x6C8	3	MBIST Control Register
STCU	MB_CTRL51	0x6CC	3	MBIST Control Register
STCU	MB_CTRL52	0x6D0	3	MBIST Control Register
STCU	MB_CTRL53	0x6D4	3	MBIST Control Register
STCU	MB_CTRL54	0x6D8	3	MBIST Control Register
STCU	MB_CTRL55	0x6DC	3	MBIST Control Register
STCU	MB_CTRL56	0x6E0	3	MBIST Control Register
STCU	MB_CTRL57	0x6E4	3	MBIST Control Register
STCU	MB_CTRL58	0x6E8	3	MBIST Control Register
STCU	MB_CTRL59	0x6EC	3	MBIST Control Register
STCU	MB_CTRL60	0x6F0	3	MBIST Control Register
STCU	MB_CTRL61	0x6F4	3	MBIST Control Register
STCU	MB_CTRL62	0x6F8	3	MBIST Control Register
STCU	MB_CTRL63	0x6FC	3	MBIST Control Register
STCU	MB_CTRL64	0x700	3	MBIST Control Register
STCU	MB_CTRL65	0x704	3	MBIST Control Register
STCU	MB_CTRL66	0x708	3	MBIST Control Register
STCU	MB_CTRL67	0x70C	3	MBIST Control Register
STCU	MB_CTRL68	0x710	3	MBIST Control Register
STCU	MB_CTRL69	0x714	3	MBIST Control Register
STCU	MB_CTRL70	0x718	3	MBIST Control Register
STCU	MB_CTRL71	0x71C	3	MBIST Control Register
STCU	MB_CTRL72	0x720	3	MBIST Control Register
STCU	MB_CTRL73	0x724	3	MBIST Control Register

Table continues on the next page...

Table 79-1. Protected registers (continued)

Module	Register	Reg offset	Protect size[1]	Description
STCU	MB_CTRL74	0x728	3	MBIST Control Register
STCU	MB_CTRL75	0x72C	3	MBIST Control Register
STCU	MB_CTRL76	0x730	3	MBIST Control Register
STCU	MB_CTRL77	0x734	3	MBIST Control Register
STCU	MB_CTRL78	0x738	3	MBIST Control Register
STCU	MB_CTRL79	0x73C	3	MBIST Control Register
STCU	MB_CTRL80	0x740	3	MBIST Control Register
STCU	MB_CTRL81	0x744	3	MBIST Control Register
STCU	MB_CTRL82	0x748	3	MBIST Control Register
STCU	MB_CTRL83	0x74C	3	MBIST Control Register
STCU	MB_CTRL84	0x750	3	MBIST Control Register
STCU	MB_CTRL85	0x754	3	MBIST Control Register
STCU	MB_CTRL86	0x758	3	MBIST Control Register
STCU	MB_CTRL87	0x75C	3	MBIST Control Register
STCU	MB_CTRL88	0x760	3	MBIST Control Register
STCU	MB_CTRL89	0x764	3	MBIST Control Register
STCU	MB_CTRL90	0x768	3	MBIST Control Register
STCU	MB_CTRL91	0x76C	3	MBIST Control Register
WKPU	NCR	0x08	3	NMI_0 configuration
WGM	CTRL	0x00	3	WGM Control Register
WGM	CTRL_1	0x04	3	WGM Control Register 1
WGM	INT_EN	0x20	1	WGM Interrupt Enable Register

1. 3 = 32 bit, 2 = 16 bit, and 1 = 8 bit

NOTE

Register protection is implemented considering 8bytes payload size for CAN_0 and CAN_2.

79.2 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE/AIPS-Lite). This is shown in the following figure.

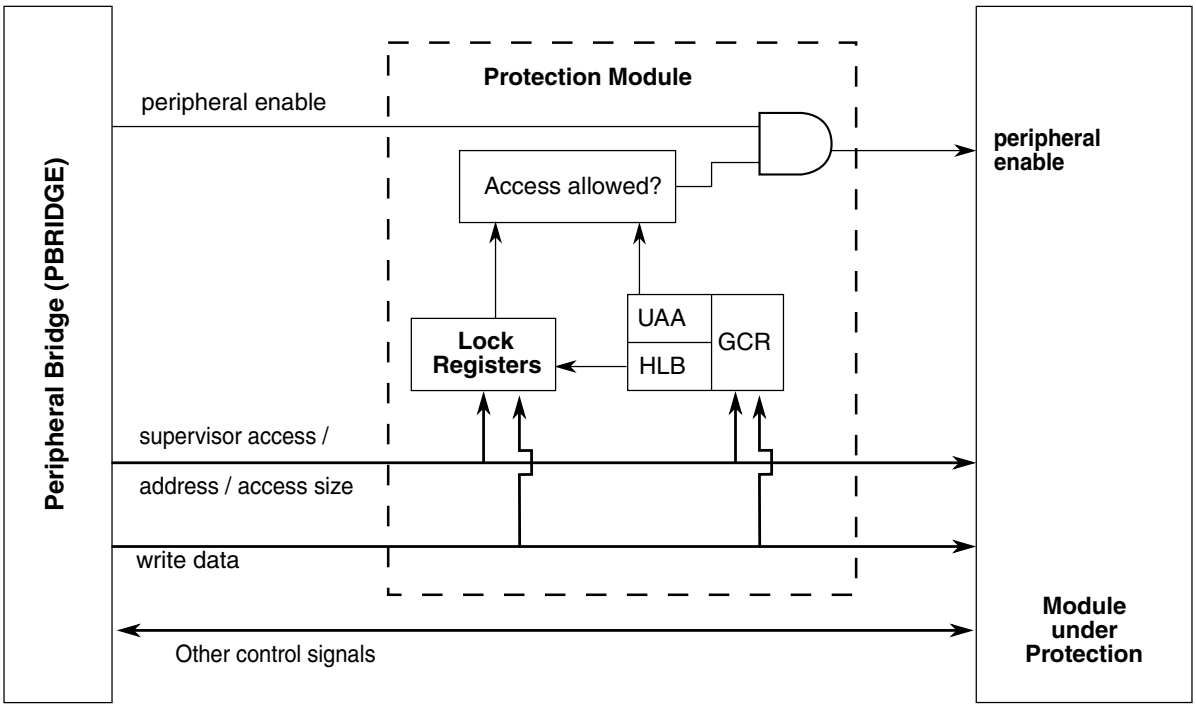


Figure 79-1. REG_PROT Block Diagram

79.3 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

79.4 Modes of operation

The Register Protection module is operable when the module under protection is operable.

79.5 External signal description

There are no external signals.

79.6 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in the following figure.

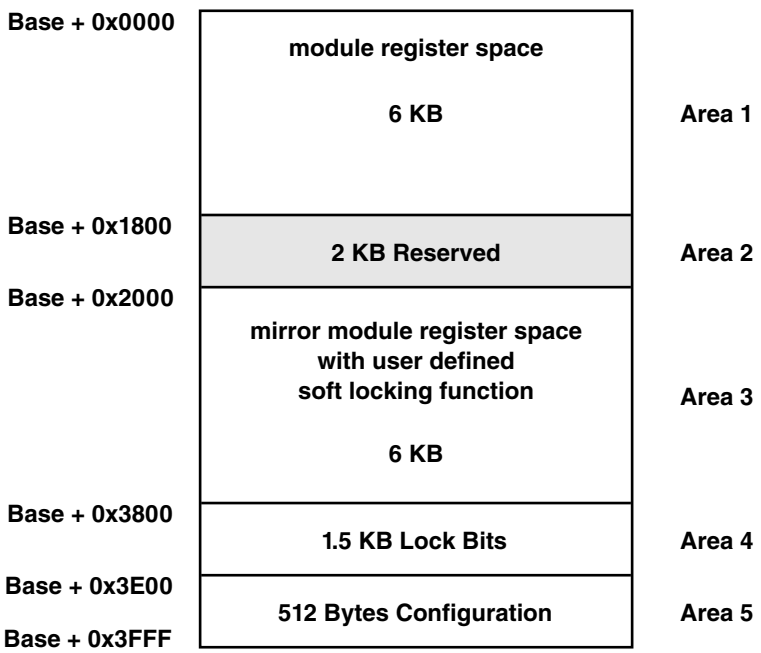


Figure 79-2. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

79.6.1 Memory map

Following is the memory map for a module which is protected via a REG_PROT module.

Table 79-2. Generic Protected Module Memory Map

Offset	Use
0x0000	Module Register 0 (MR0)
0x0001	Module Register 1 (MR1)
0x0002	Module Register 2 (MR2)
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)
0x1800 - 0x1FFF	Reserved
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 -Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143
0x3E00 - 0x3FFB	Reserved
0x3FFC	Global Configuration Register (GCR)

Note

Reserved registers in area #1 will be handled as specified in the protected module's documentation.

79.6.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

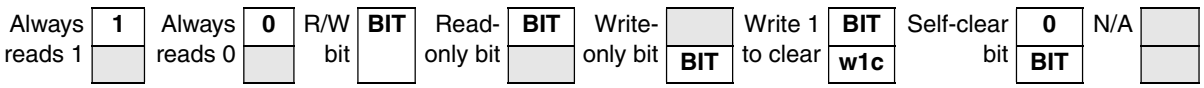


Figure 79-3. Register description

79.6.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

79.6.2.2 Module register and set soft lock bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in the Soft Lock Bit Register section.

79.7 Memory map and registers

REG_PROT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3800	Soft Lock Bit Register n (REG_PROT_SLBRn)	8	R/W	00h	79.7.1/3970
3FFC	Global Configuration Register (REG_PROT_GCR)	32	R/W	0000_0000h	79.7.2/3972

79.7.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area #1.

Table 79-3. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

NOTE

- As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.
- Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3800h offset = 3800h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0	0	SLB0	SLB1	SLB2	SLB3
Write	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

REG_PROT_SLBRn field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB) WE0 enables writing to SLB0

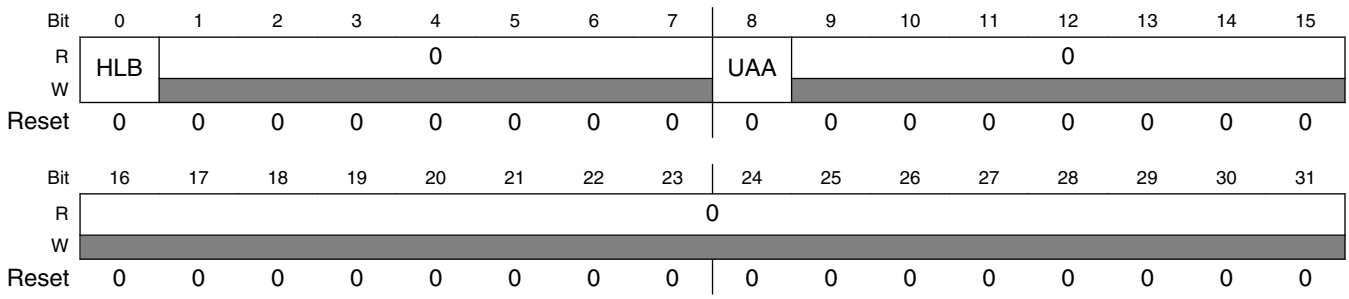
Table continues on the next page...

REG_PROT_SLBRn field descriptions (continued)

Field	Description
	1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB) WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB) WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB) WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MRn register SLB0 can block accesses to MR[n *4 + 0] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MRn register SLB1 can block accesses to MR[n *4 + 1] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MRn register SLB2 can block accesses to MR[n *4 + 2] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MRn register SLB3 can block accesses to MR[n *4 + 3] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable

79.7.2 Global Configuration Register (REG_PROT_GCR)

Address: 0h base + 3FFCh offset = 3FFCh



REG_PROT_GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit This register cannot be cleared once it is set by software. It can only be cleared by a system reset. **NOTE:** Access in User Mode is read/write if UAA is set to 1 otherwise it is read-only; in Supervisor Mode access is read/write. 1 All SLB bits are write protected and can not be modified. 0 All SLB bits are accessible and can be modified.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 UAA	User Access Allowed. Controls the User and Supervisor mode access to registers of the module under protection, as well as the REG_PROT_GCR and REG_PROT_SLBR*n*. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can be written in supervisor mode only. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

79.8 Functional description

The following sections describe the functional characteristics of the Register Protection module.

79.8.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses (peripheral registers) that are protected there are SLBRn.SLBm bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBRn.SLBm bit is always 0b0 no matter what software is writing to.

79.8.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBRn.SLBm bit needs to be changed. This can be done using the following methods:

- Modify the SLBRn.SLBm directly by writing to area #4
- Set the SLBRn.SLBm bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

79.8.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBRn.SLBm bit has a mask bit SLBRn.WEm which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

The following figure shows two modification examples. In the left example there is a write access to the SLBRn register specifying a mask value which allows modification of all SLBRn.SLBm bits. The example on the right specifies a mask which only allows modification of the bits SLBRn.SLB[3:1].

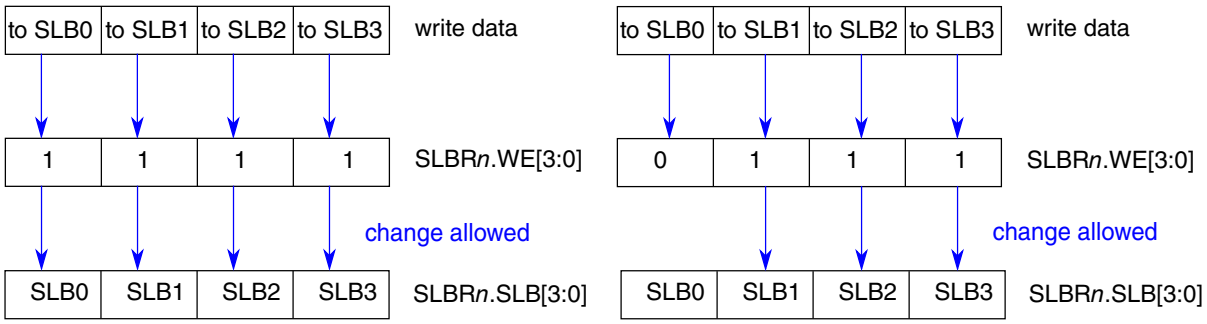


Figure 79-4. Change lock settings directly via Area #4

The previous figure shows four registers that can be protected 8-bit wise. The following figures show registers with 16- and 32-bit protection respectively:

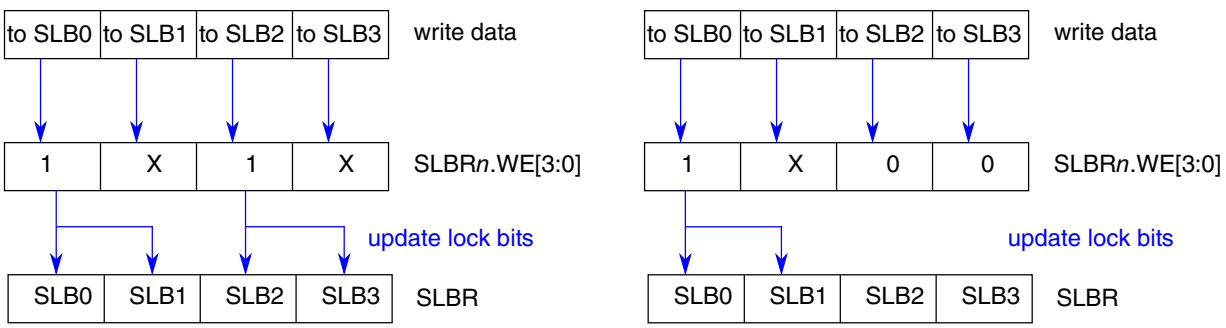


Figure 79-5. Change lock settings for 16-bit protected addresses

On the right side of the previous figure it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of the previous figure the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

The next figure shows a 32-bit wise protected register. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

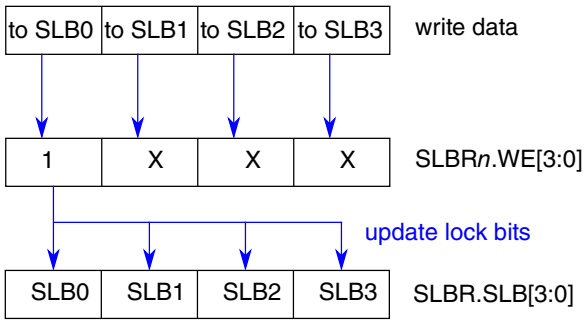


Figure 79-6. Change lock settings for 32-bit protected addresses

The following figure shows a mixed protection size configuration:

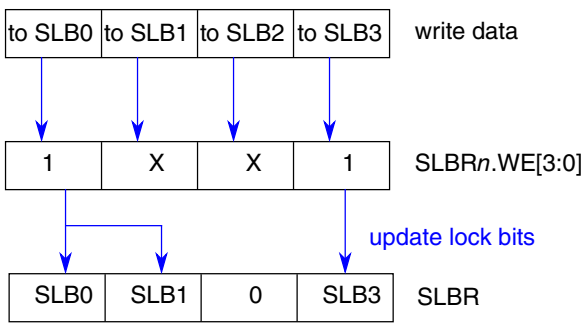


Figure 79-7. Change lock settings for mixed protection

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

79.8.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. For example:

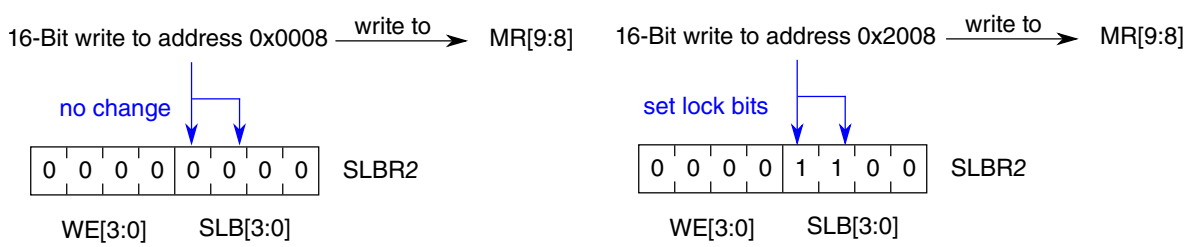


Figure 79-8. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (see the left part of Figure 79-5).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 79-5).

The following figure shows an example where some addresses are protected and some are not:

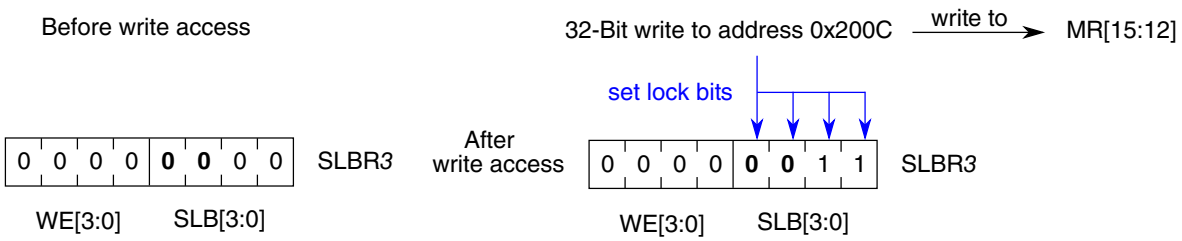


Figure 79-9. Enable locking for protected and unprotected addresses

In the previous figure addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

79.8.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings directly via Area #4 and Enable locking via mirror module space (Area #3) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

79.8.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 79-2

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.

3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

Chapter 80
Password and Device Security Module (PASS)

80.1 Chip-specific PASS information

80.1.1 Chip-specific Password and Device Security Module (PASS) information

The PASS module is used to implement password-based read and write protection for flash blocks. Up to four levels of password protection can be implemented for each flash block.

NOTE

The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See DCF tables for detailed information about DCF clients for the PASS (Excel file) attached to this document.

Each password group defined in the DCF records has a set of four LOCK*n* registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the Flash Block Assignment tables of the device.

See the Flash block assignment table (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the PASS LOCK Reg tab.

NOTE

Users should ensure that no program or erase operations are occurring on any memory partitions shared with the UTEST block before initiating a password challenge. This can be monitored through C55FMC_MCR[PGM] and C55FMC_MCR[ERS] fields in C55FMC memory map.

80.1.2 Flash Test Mode protection

In Failure Analysis Lifecycle, Flash Test Mode entry is protected by combination of below two inputs.

1. DCF_FTMP (Refer to attached DCF Sheet)
2. TEST MODE ENABLE_DCF_CLIENT

Table 80-1. Flash Test Mode protection

DCF_FTMP	TEST MODE ENABLE DCF CLIENT	Flash TEST MODE ACCESS
0	Don't care	YES (NO Test mode enable PASSWORD REQUIRED)
1	0	Test Mode not enabled.
1	1	YES (Test mode enable PASSWORD REQUIRED)

80.2 Introduction

80.2.1 Overview

The Password and device security (PASS) module shown in the following figure, receives password challenges and determines their validity. It also maintains the device security and access states.

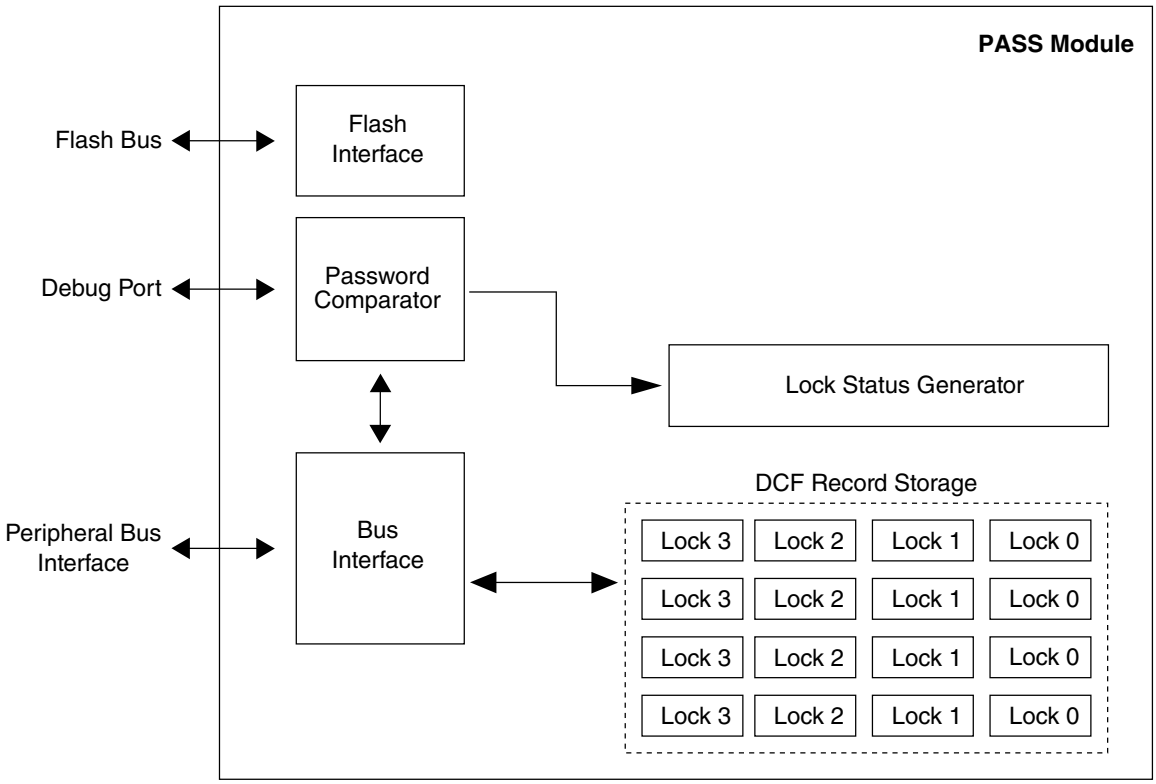


Figure 80-1. PASS module block diagram

80.2.2 Features

The PASS includes these distinctive features:

- Life Cycle status
- Password comparison
- JTAG password comparison
- Test mode enable Configuration and Password Comparison
- Flash read, write, and erase control

80.2.3 Modes of operation

The PASS operates identically in all system modes.

80.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in flash UTEST region.

The following table shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

NOTE

Non-implemented registers generate bus aborts if enabled.

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write '0' to these bits; they always read '0'.

PASS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Life Cycle Status Register (PASS_LCSTAT)	32	R	See section	80.3.1/3983
8	Challenge Selector Register (PASS_CHSEL)	32	R/W	0000_0000h	80.3.2/3985
10	Challenge Status Register (PASS_CSTAT)	32	R	0000_0000h	80.3.3/3985
20	Challenge Input Register (PASS_CIN0)	32	W (always reads 0)	0000_0000h	80.3.4/3986
24	Challenge Input Register (PASS_CIN1)	32	W (always reads 0)	0000_0000h	80.3.4/3986
28	Challenge Input Register (PASS_CIN2)	32	W (always reads 0)	0000_0000h	80.3.4/3986
2C	Challenge Input Register (PASS_CIN3)	32	W (always reads 0)	0000_0000h	80.3.4/3986
30	Challenge Input Register (PASS_CIN4)	32	W (always reads 0)	0000_0000h	80.3.4/3986
34	Challenge Input Register (PASS_CIN5)	32	W (always reads 0)	0000_0000h	80.3.4/3986
38	Challenge Input Register (PASS_CIN6)	32	W (always reads 0)	0000_0000h	80.3.4/3986
3C	Challenge Input Register (PASS_CIN7)	32	W (always reads 0)	0000_0000h	80.3.4/3986

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG0)	32	R/W	See section	80.3.5/3986
104	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG0)	32	R/W	See section	80.3.6/3988
108	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG0)	32	R/W	See section	80.3.7/3989
10C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG0)	32	R/W	See section	80.3.8/3990
110	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG1)	32	R/W	See section	80.3.5/3986
114	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG1)	32	R/W	See section	80.3.6/3988
118	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG1)	32	R/W	See section	80.3.7/3989
11C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG1)	32	R/W	See section	80.3.8/3990
120	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG2)	32	R/W	See section	80.3.5/3986
124	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG2)	32	R/W	See section	80.3.6/3988
128	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG2)	32	R/W	See section	80.3.7/3989
12C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG2)	32	R/W	See section	80.3.8/3990
130	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG3)	32	R/W	See section	80.3.5/3986
134	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG3)	32	R/W	See section	80.3.6/3988
138	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG3)	32	R/W	See section	80.3.7/3989
13C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG3)	32	R/W	See section	80.3.8/3990

80.3.1 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle input from external blocks.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	TME	0												
W																
Reset	*	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*

* Notes:
- LIFE field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- CNS field: Reset value depends on the Censorship DCF client

PASS_LCSTAT field descriptions

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA. The bit can be set (or cleared) via DCF Record in UTest Flash. **NOTE:** If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. (In life cycle FA, the JUN bit will always be 0.) Unsuccessful transmission clears this bit.
2 TME	Test Mode enable. This field indicates whether the test mode DCF Client is set to enable test mode in FA. 0 Test mode cannot be enabled 1 Test mode can be enabled with test mode password
3–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

80.3.2 Challenge Selector Register (PASS_CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.

When a master writes this register, its ID sets the CMST field of the Challenge Status register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														GRP	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CHSEL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

80.3.3 Challenge Status Register (PASS_CSTAT)

This register provides status information for the password challenge.

Only the master shown in this register can access the CINn registers - access by other masters result in a comparison fail regardless of the values written.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												CMST			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CSTAT field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CMST	ID of the master which has last written the CHSEL register

80.3.4 Challenge Input Register (PASS_CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7 in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed. In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as clears the PGL bit of that register.

All accesses to these registers need to be made by the bus master that wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

Address: 0h base + 20h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	PW32																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CIN*n* field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

80.3.5 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

LOCK0_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', indicating that a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	Reserved	LOWLOCK													
W																
Reset	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- MIDLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

- LOWLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- TSLOCK field: If no DCF records have been written to change the TS bit state, the bit will reset to ‘1’.

PASS_LOCK0_PG*n* field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 Reserved	This field is reserved.
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

80.3.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK1_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is '1', which indicates a block is locked. If no DCF records have been written to change the lock state, the bits will reset to '1'.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of Flash blocks.

CAUTION

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- HIGHLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

PASS_LOCK1_PG*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

80.3.7 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to OEM Production or older.

NOTE

Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in this section.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is ‘1’, which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L_256LCK																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- L_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit resets to ‘1’.

PASS_LOCK2_PG*n* field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

80.3.8 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK3_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

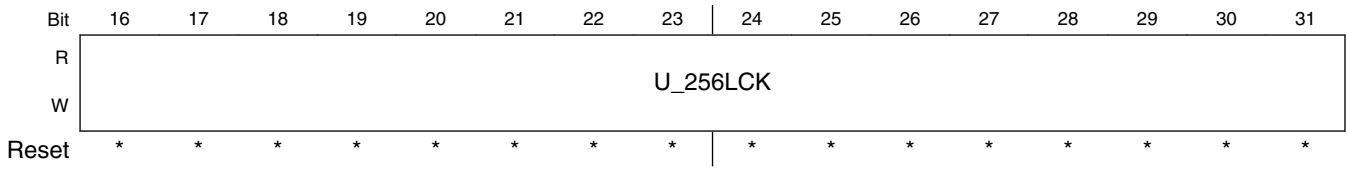
Once a censored device enters the Failure Analysis life cycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle.

NOTE

Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0			RL4	RL3	RL2	RL1	RL0
W																
Reset	*	*	*	0	1	1	1	1	0	0	0	*	*	*	*	*



* Notes:

- U_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- RL0 field: If no DCF records have been written to change the RL0 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL1 field: If no DCF records have been written to change the RL1 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL2 field: If no DCF records have been written to change the RL2 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL3 field: If no DCF records have been written to change the RL3 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL4 field: If no DCF records have been written to change the RL4 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- MO field: If no DCF records have been written to change the MO bit lock state, the bit will reset to ‘1’.
- DBL field: If no DCF records have been written to change the DBL bit lock state, the bit will reset to ‘1’.
- PGL field: Reset value depends on life cycle. If life cycle is Customer Delivery or earlier, the value will be 0, otherwise 1.

PASS_LOCK3_PG*n* field descriptions

Field	Description
0 PGL	Password Group Lock. This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF load from Flash. This bit would be automatically locked once Flash Scanning is finished after functional reset. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)
1 DBL	Debug Interface Lock. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from DCF records in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, it can only be updated by writing to it once the passgroup has been unlocked via the PASS Password challenge mechanism.

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
	0 Debug interface is unlocked. Accessed to all JTAG clients is allowed 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only. 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MSTR	Master Access. Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 RL4	Read Lock Regison 4 lock bit. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 4 is not protected. 1 Region 4 is protected.
12 RL3	Region 3 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 3 is not protected. 1 Region 3 is protected.
13 RL2	Region 2 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Region 1 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 1 is not protected. 1 Region 1 is protected.
15 RL0	Region 0 is protected.

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
	The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 0 is not protected. 1 Region 0 is protected.
16–31 U_256LCK	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

80.4 DCF clients

The following table shows the DCF clients for the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

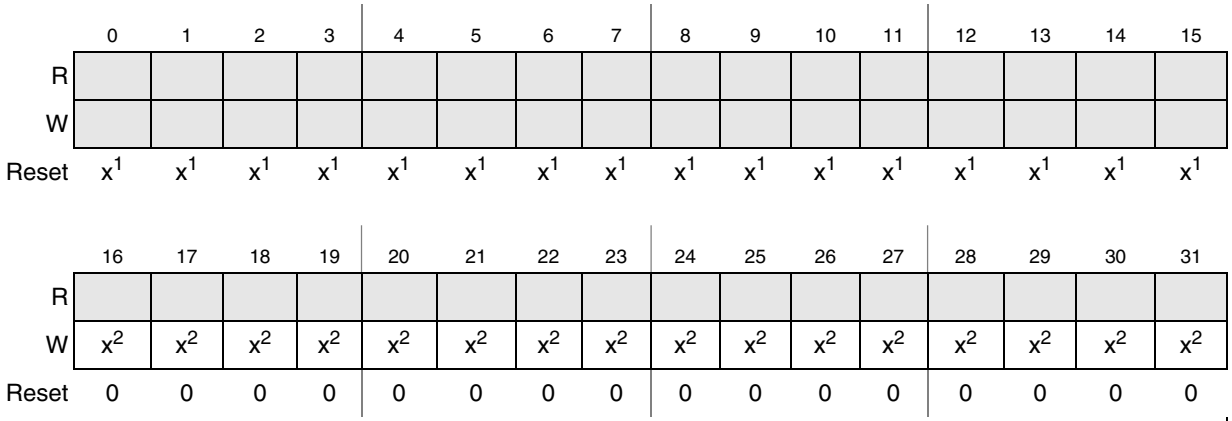
The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual or in DCF sheet attached to this document.

Table 80-2. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	Censorship
Test mode enable					
None	Test mode enable	None	0x0000_0001	0x00E0	Test mode enable DCF Client

80.4.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 This bit is not defined

2 This value is loaded by DCF Record during reset

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

80.4.2 Test mode enable DCF Client

This DCF client is used to configure the password protected Test mode enable feature. See Test mode enable for details.

The Test mode DCF Client is not directly readable in software, but the state of the Test mode bit can be read in the LCSTAT register (Life Cycle Status Register (PASS_LCSTAT)).

DCF Client Address: 0x00E0 Test Mode Enable — Access:DCF Record Read Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																TM[2]
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	1

[1] This bit is not defined

[2] This value is loaded by DCF Record during reset

Figure 80-2. Test mode enable DCF Client

Table 80-3. Test mode enable register field descriptions

Field	Description
0-30	Reserved
31 TM	Test mode enable '0' = Test mode cannot be enabled '1' = Test mode can be enabled with a password

80.5 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

80.5.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access and flash memory read protection.

The Debug Interface access is controlled by the CNS and LIFE bits of the LCSTAT register (Life Cycle Status Register (PASS_LCSTAT)) according to the Debug Interface Access truth table. The flash memory read protection is also controlled by the CNS and LIFE bits of the LCSTAT register according to the Flash memory read protection truth table.

The CNS bit is set or cleared by executing a DCF record that writes to the Censorship DCF client (See Censorship). The LIFE bits of the LCSTAT register are controlled by the SSCM using user programmed values from specific locations in the UTEST flash.

80.5.2 Debug Interface Access

The debug interface is fully enabled in Customer Delivery life cycle. But from OEM Production life cycle, it can be blocked using censorship and unlocked by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 registers (See Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)). Both these debug enabling options are further conditioned by life cycle.

The table below describes all options for enabling/blocking Debug Interface access

Table 80-4. Debug Interface Enable Truth Table

Input Conditions				Outcome
Lifecycle[1]	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Not censored	Don't Care	Don't Care	Enabled
	Censored	Unlocked (DBL=0)	Don't Care	Enabled
		Locked (DBL=1)	Matched	Enabled
			Not Matched	Blocked
In Field	Not censored	Don't Care	Don't Care	Enabled
	Censored	Unlocked (DBL=0)	Don't Care	Enabled
		Locked (DBL=1)	Matched	Enabled
			Not Matched	Blocked
Failure Analysis	Don't Care	Don't Care	Don't Care	Enabled

1. Life cycle is governed by life cycle management module (for, example SSCM)

80.5.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled. The password matching is done only after the chip has come out of reset PHASE3. The Debug Interface Access stays enabled until the next destructive reset.

80.5.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Life Cycle, Censorship and whether the Debug Interface access is enabled or blocked. The Flash memory read protection truth table describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The flash memory is also read protected when the life cycle status is Failure Analysis. This life cycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

Table 80-5. Flash Memory Read Protection Truth Table

INPUTS			OUTPUTS	
Lifecycle	**Censorship**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
Customer Delivery	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Readable	Readable
	Censored	Locked (1)	Read Blocked	Read Blocked
		Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Readable	Readable
	Censored	Locked (1)	Readable	Read Blocked
		Locked (0)	Readable	Readable

80.5.4 Flash Memory Write and Erase Protection

This section describes write and erase permissions for flash memory. From Power ON to SSCM Done, all regions of flash memory are in locked state. After Power ON, SSCM scans flash memory for valid DCF records. These DCF records carry program and erase protection settings for flash memory regions. SSCM module finds the settings and load

them into PASS configuration via the DCF path. Once SSCM Done is asserted, the new configuration loaded by SSCM into PASS come into force and unlocks different regions of flash for program and erase.

To enable write and erase access of a particular region of flash memory, add corresponding DCF records in the flash memory once chip comes out of reset (i.e. SSCM Done). User can also change the settings by providing valid password to the PASS module, and by modifying the settings of following registers bits fields:

- PASS_LOCK0_PGn[MIDLOCK]
- PASS_LOCK0_PGn[LOWLOCK]
- PASS_LOCK1_PGn[HIGHLOCK]
- PASS_LOCK2_PGn[L_256LCK]
- PASS_LOCK3_PGn[U_256LCK]

See chip-specific PASS information for mapping of the above mentioned bits fields.

In the PASS module there is no individual access protection for Write or Erase access types, but Write and Erase accesses summated.

80.5.5 Test mode enable

Test mode can be used for testing different features of Flash and internal blocks. These features can be used in Failure analysis life cycle. Following are the guidelines for customers:

- During customer assembly, customer programmes DCF Record in UTEST to write test mode DCF client with 1.
- During customer assembly, the customer programmes the test mode enable Password into Flash
- The customer maintains a system for retrieving these passwords for the life of the module
- If the part is returned to JDP for FA, the customer must provide the test mode enable password that was originally programmed into the device.
- If the password is lost or irretrievable, the customer must accept that FA cannot be done.

NOTE

Usage of this case causes restrictions on failure analysis.

80.5.6 Test mode enable password

The Test mode enable password is a password entered by the manufacturer via JTAG interface, during device FA, to gain access to Flash Test Mode. Flash Test Mode is required to perform Vt Distribution analysis and other tests on an MCU. This password is only functional during FA life cycle. During other life cycles, Flash Test Mode is either available; completely blocked; or requires the Retest Password. Flash blocks that are assigned as sealed during production are locked against any form of Flash test, including reading, erasing and Vt Distribution analysis.

80.5.6.1 Test mode enable password generation

The Test mode enable password is created by the customer during their module assembly process. It must be programmed into UTEST Flash while in Customer Delivery life cycle.

When an MCU is returned to manufacturer for FA, it must be in FA life cycle and the password must also be provided to the manufacturer to allow access to Flash Test Mode. Therefore the customer should develop a process for managing Test mode enable passwords for the life of the module.

80.5.6.2 Test mode enable password storage

The test mode enable password is 224 bits (28 bytes) wide and is to be stored in UTEST Flash at a fixed location. See the device reference manual for the address of the test mode password.

The password region of UTEST Flash is readable only when in Customer Delivery life cycle. In any life cycle after this (including FA life cycle), this region is read protected.

80.5.6.3 Test mode enable password Challenge entry

The Test mode enable password challenge in entered by an external tool to the password register in JTAGC

As the Test mode enable password is 224 bits; and the JTAGC password register is 256, only the least significant 224 bits of the register are used.

80.5.6.4 Test mode enable password compare

The Test mode password challenged, is compared against the password stored in UTEST Flash. Only the least significant 224 bits are compared as the most significant 32 bits of the Flash page contain different data not related to Test mode enable.

If the password matches, Flash Test Mode access is allowed until the next POR reset.

There are no passwords that the hardware will detect as illegal. All 0 and all 1 are valid passwords.

80.6 Initialization/application information

80.6.1 Reset

The reset state of each individual bit is shown within the Register Description section. However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

80.6.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 80-6. CS/ADDR Values for Lock Registers

Register	DCF Command Word
PW Group 0	
Lock0	0xxxxx[1] _0100
Lock1	0xxxxx[1] _0104
Lock2	0xxxxx[1] _0108
Lock3	0xxxxx[1] _010C
PW Group 1	
Lock0	0xxxxx[1] _0110
Lock1	0xxxxx[1] _0114
Lock2	0xxxxx[1] _0118
Lock3	0xxxxx[1] _011C
PW Group 2	
Lock0	0xxxxx[1] _0120

Table continues on the next page...

Table 80-6. CS/ADDR Values for Lock Registers (continued)

Register	DCF Command Word
Lock1	0xxxxx[1] _0124
Lock2	0xxxxx[1] _0128
Lock3	0xxxxx[1] _012C
PW Group3	
Lock0	0xxxxx[1] _0130
Lock1	0xxxxx[1] _0134
Lock2	0xxxxx[1] _0138
Lock3	0xxxxx[1] _013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See chip specific PASS information for the Client Select assignment

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in Table 80-7.

Table 80-7. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0
0x0C	0xxxxx_0110	Set MIDLOCK to 0xFF
0x10	0xFFFF_FFFF	Non-Programmed Addresses
0x14	0xFFFF_FFFF	
…	…	

Chapter 81
Tamper Detection Module (TDM)

81.1 Chip-specific TDM information

81.1.1 Chip-specific TDM information

Flash block assignment tables contains the TDM configuration registers implementation in this device.

See the device Flash block assignment (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the TDR and OTP LOCK Reg tab.

81.2 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that requires software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prohibit* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a "diary". At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 4 TDRs can be defined via DCF records.

The following figure shows the block diagram of the TDM and its interface to other system components.

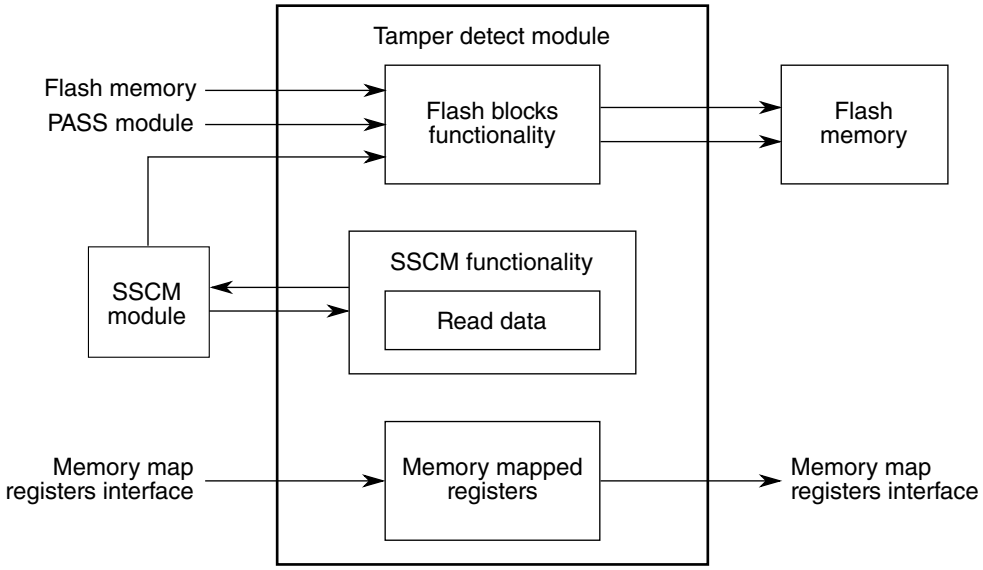


Figure 81-1. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module via the DCF bus. Among the data are:

- OTP (One Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information
- Software tamper region disable

The TDM module controls the flash blocks for erase operations and sets them as OTP.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

NOTE

The pass module provides the control for erase operation for each flash block as well as for each region of flash memory, and the PASS settings have priority over the settings of TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

81.3 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 4 parts (TDRs), each consisting of 256 64-bit records.
  - The diary block should be assigned as OTP region in flash.
  - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock DCF clients for each flash memory TDR defined.
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[1]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory mapped registers interface for reading via software of specific registers.

81.4 External signal description

There are no external signals driving or being driven by the TDM.

81.5 DCF clients

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/ erase activity
- Permanently disable one or more Tamper Regions
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

1. The Tamper Region Override DCF record is a "write one only" DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

Table 81-1. DCF clients

DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
Software Tamper Region Override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR. • n specifies a TDR • m specifies the flash blocks as per the LOCK registers of the flash memory.	write 0 to 1 only

81.5.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. The following diagram depicts this client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0000 Access: DCF write

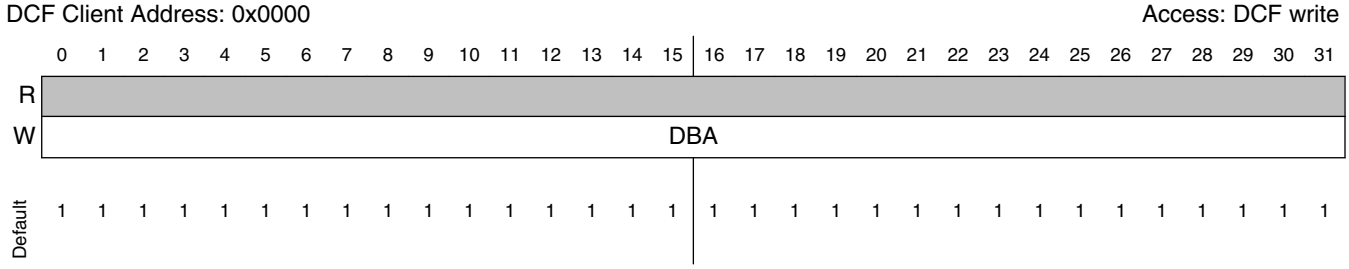


Table 81-2. Diary Base Address (DBA) DCF client field description

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **NOTE:** Diary Base Address must be on a 16 KB boundary.

81.5.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. The following figure shows the diagram for this register.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE3	TOE2	TOE1	TOE0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81-3. Tamper Region Override (TO) DCF client field descriptions

Field	Description
0–27 Reserved	This field is reserved and should be written with all zeroes when programming the TOE fields.
28–31 TOE3 to TOE0	TDR Override Enable. Each TOE*n* field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written. 1 Diary override. The blocks assigned to this TDR can be erased WITHOUT writing to the diary.

81.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client

This DCF client holds information to disable the software mechanism to override the tamper detect regions.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0002 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STO_DIS3	STO_DIS2	STO_DIS1	STO_DIS0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81-4. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions

Field	Description
0–27 Reserved	This field is reserved and should be written with all zeroes when programming the STO_DIS fields.
28–31 STO_DIS3 to STO_DIS0	Software TDR Override Disable TDR*n*. Each STO_DIS*n* field, where *n* represents the TDR number, holds information to disable the software mechanism to override TDR*n*. 0 Software tamper detect override mechanism supported 1 Software tamper detect override mechanism disabled. Any attempt to override TDR*n* by loading a service key value in STO_KEY*n* register is ignored.

81.5.4 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. The following diagram depicts this client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWOTPEN[11:0]												MIDOTPEN[15:0]															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81-5. One Time Programmable Enable (OTPEN0) DCF client

Field	Description
0–3 Reserved	This field is reserved and should be written with all zeroes when writing to this client.
4–15 LOWOTPEN	Low Block OTP Enable. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16–31 MIDOTPEN	Mid Block OTP Enable. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

81.5.5 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0009 Access: DCF write

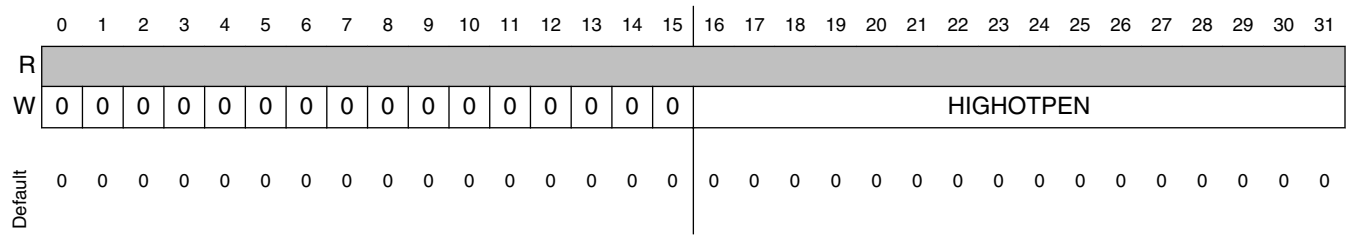


Table 81-6. One Time Programmable Enable (OTPEN1) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHOTPEN	High Block OTP Enable. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

81.5.6 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000A Access: DCF write

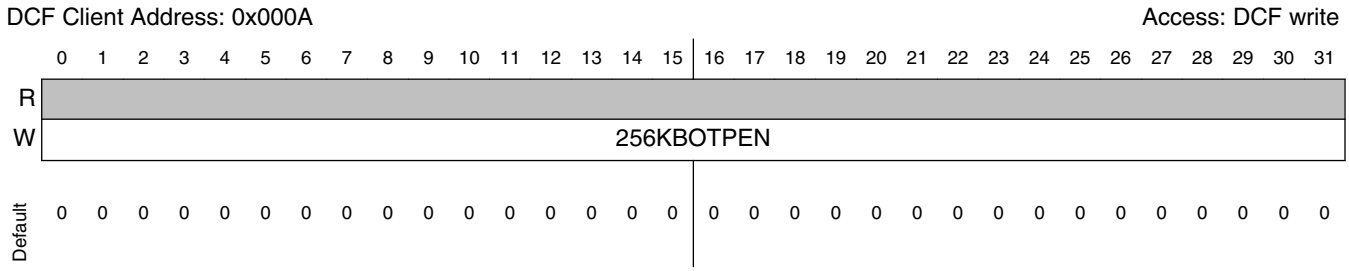


Table 81-7. One Time Programmable Enable (OTPEN2) DCF client field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

81.5.7 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK3 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000B — Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256KBOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81-8. One Time Programmable Enable (OTPEN3) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in the upper range of 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

81.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 4). This DCF client holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK0 register in the flash memory. The following diagram depicts the TDRn_LOCK0 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 0) Access: DCF write

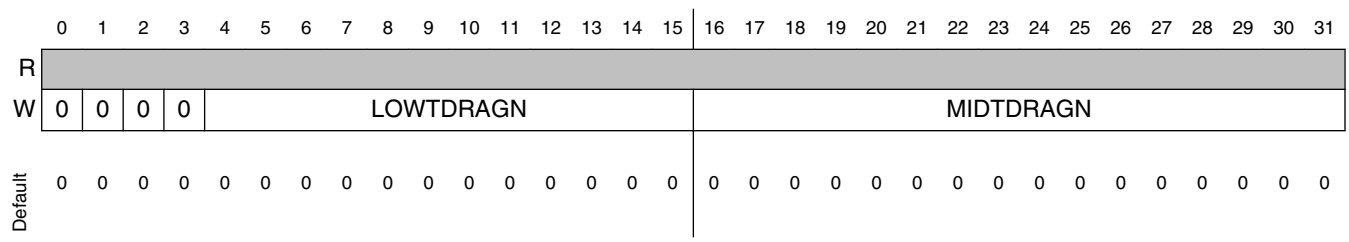


Table 81-9. Tamper Region Assignment DCF client (TDRn_LOCK0)

Field	Description
0–3 Reserved	This field is reserved and should be written with zeroes when writing to this client.
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

81.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 4). This DCF client holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK1 register in the flash memory. The following diagram depicts the TDRn_LOCK1 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 1) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHTDRAGN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81-10. Tamper Region Assignment DCF client (TDRn _LOCK1)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHTDRAGN	High Block Tamper Region Assignment. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

81.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 4). This DCF client holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK2 register in the flash memory. The following diagram depicts the TDRn_LOCK2 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 2) Access: DCF write

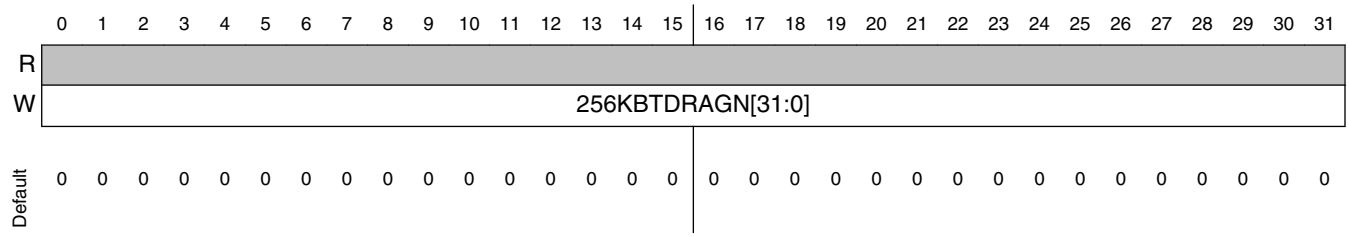


Table 81-11. Tamper Region Assignment DCF client (TDRn_LOCK2)

Field	Description
0–31 256KBTDRAGN[31:0]	256 KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

81.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 4). This DCF client holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK3 register in the flash memory. The following diagram depicts the TDRn_LOCK3 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 3) Access: DCF write

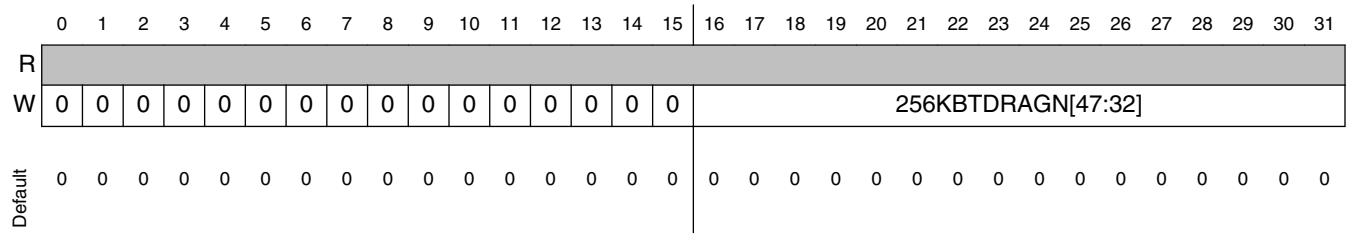


Table 81-12. Tamper Region Assignment DCF client (TDRn_LOCK3)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBTDRAGN[47:32]	256KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

81.6 Memory Map/Register Definition

TDM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	TDR Status Register (TDM_TDRSR)	32	R	See section	81.6.1/4016
4	Last Flash Programmed Address Register (TDM_LFPAR)	32	R	0000_0000h	81.6.2/4018
8	Diary Base Address (TDM_DBA)	32	R	See section	81.6.3/4018
10	Software Tamper Override Key Region (TDM_STO_KEY0)	32	R/W	0000_0000h	81.6.4/4019
14	Software Tamper Override Key Region (TDM_STO_KEY1)	32	R/W	0000_0000h	81.6.4/4019
18	Software Tamper Override Key Region (TDM_STO_KEY2)	32	R/W	0000_0000h	81.6.4/4019
1C	Software Tamper Override Key Region (TDM_STO_KEY3)	32	R/W	0000_0000h	81.6.4/4019

81.6.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, i.e., locked or unlocked, of each tamper region, i.e., TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

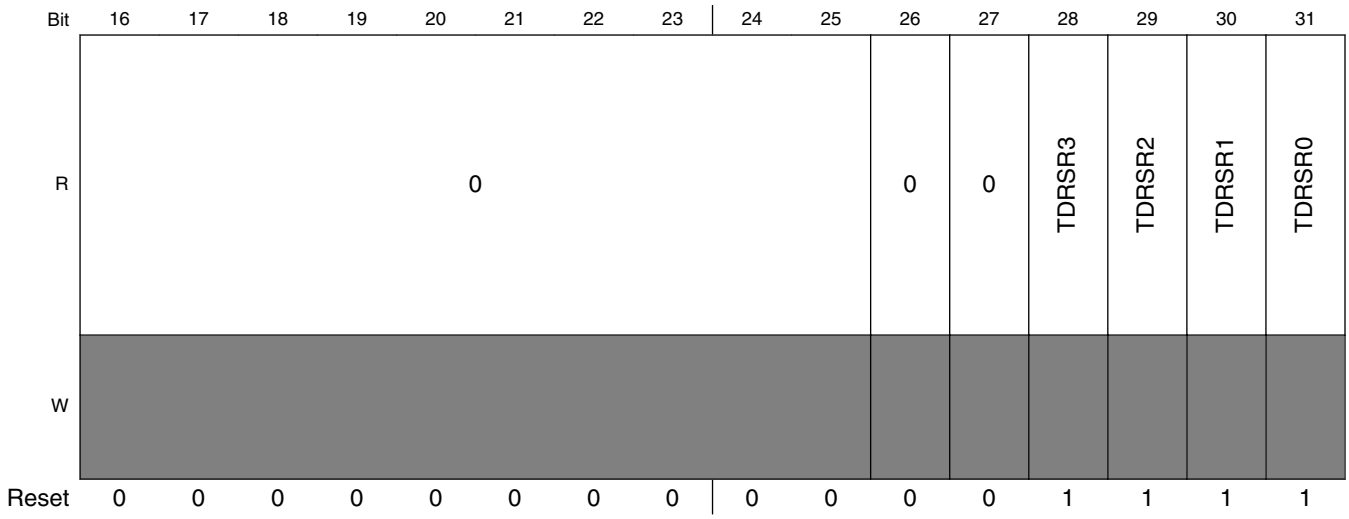
1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

NOTE

- TDRSR is a read-only register and writes have no effect on it. Write operations result in a transfer error.
- If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



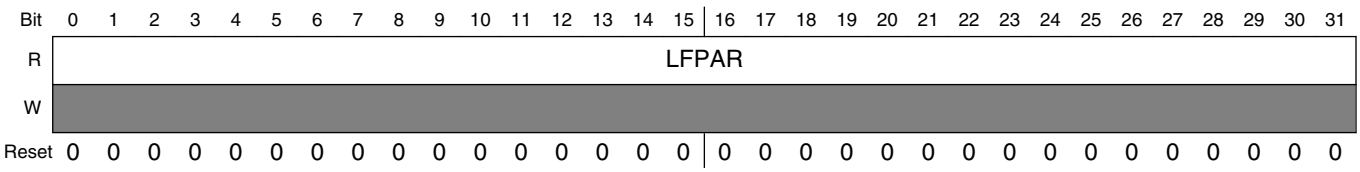
TDM_TDRSR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

81.6.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Address: 0h base + 4h offset = 4h



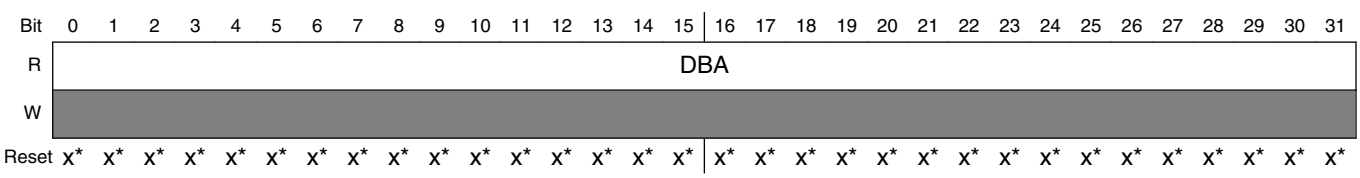
TDM_LFPAR field descriptions

Field	Description
0–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

81.6.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Address: 0h base + 8h offset = 8h



* Notes:
- The reset value of this register is set via DCF record.x = Undefined at reset.

TDM_DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. These bits represent the address of the diary location in the flash. **NOTE:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

81.6.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region n, where n denotes the Tamper Detect Region number.

Caution

1. The STO_KEYn registers can only be written once.
2. The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Address: 0h base + 10h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STO_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_STO_KEY*n* field descriptions

Field	Description
0–31 STO_KEY	Software Tamper Override Key Diary n. This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Caution:** The STO_KEYn registers can only be written once.

81.7 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

81.7.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the diary is defined by a DCF Record.

The diary is divided into 4 parts where each part consists of 256 × 64-bit records, to record up to 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 × 32-bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDR*n*_LOCK*m* register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensures that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double-word program operation, the flash module indicates to the TDM the address of the programmed double word. The TDM determines whether the programmed data was written to the correct TDR diary location and then unlocks the blocks for erase operation for the corresponding TDR. The TDM ensures that this erase operation on unlocked blocks is allowed until the completion of the erase sequence or until next desctructive reset.

When attempting to update the TDM diary and erase flash blocks, set MCR[PECIE] in the Embedded Flash Memory registers to enable interrupt notification for program and erase operations. Then clear MCR[PECIE] to service the interrupt after each program or erase operation. This sequence must be executed before attempting to update the TDM diary, and then again before attempting to erase the unlocked flash blocks.

NOTE

All blocks associated to a TDR where an erase operation is allowed can be erased all together.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

NOTE

There is no restriction on the type or format of data that can be programmed in the diary, so software must verify whether the diary is full. If a TDR's diary area is full, the tamper detect mechanism can be disabled for that TDR by creating a Tamper Region Override (TO) DCF record with the TDR's corresponding TDR Override Enable (TOE) bit set to '1'. This

allows erase operations to be performed to the TDR without any diary entry.

The setting from “override tamper region lock” DCF record (Diary) has priority over the settings in the TDRn_LOCK0 – 3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, if a block is locked by the PASS_LOCK*x*_PG*n* registers or by TDR*n*_LOCK*m*, the block is locked for erase.

NOTE

The settings in the Tamper Region Override Register have priority over the TDR*n*_LOCK*m* registers. The PASS module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via the TDM, but the same blocks are programmed to be locked for erase operation in PASS, the blocks end up being locked for erase operation.

Attempts to erase the flash without first writing the diary will be ignored by the flash. The contents in the flash block will remain unchanged and MCR[PEG] will be asserted, indicating the operation was successful and the contents of the block are properly protected from the erase operation.

81.7.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 x 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDR*n*_LOCK*m* where *n* is the number of a tamper region, that is, 0- 4 and *m* varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- ’1’ means that the corresponding flash block is assigned to this TDR
- ’0’ means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They can not be read nor written. They can only be initialized by DCF record within the UTest flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

NOTE

No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

81.7.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes to honor OTP restrictions. Therefore, each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

NOTE

To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

81.7.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore, each diary region limits the erases of the assigned blocks to 256 erases.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once and the default bit state is ‘1’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 4 TDRs, 8 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xx00_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

81.7.5 TDR lock status

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as "TDR Erase Blocked". See TDR Status Register (TDM_TDRSR) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 4 status bits can be read through the TDRSR. See TDR Status Register (TDM_TDRSR). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See Last Flash Programmed Address Register (TDM_LFPAR).

81.7.6 TDR override

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to ‘0’ in each 64 bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

The application can also override the diary by loading a service key to the STO_KEY*n* program-visible register, where n denotes the tamper detect region. By loading the value 0x55AA5A5A into STO_KEYn register, the corresponding tamper detect region is overridden. This software override mechanism is disabled if the STO_DISn field of the STO_DIS DCF Client is set.

81.7.7 One time programmable (OTP)

81.7.7.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next functional reset.

Chapter 82
Cryptographic Services Engine (CSE)

82.1 Chip-specific CSE information

82.1.1 Overview

This chip has one instance of the CSE module. When enabled, the CSE module:

- Can execute the chip's secure boot process.
- Has exclusive access to the flash memory blocks mapped to the 16 Kb HIGH blocks of the C55FMC_LOCK1 register, PASS_LOCK1_PGn registers, and TDRn_LOCK1 DCF client. The system MPU is automatically configured to prevent other bus masters from interfering with CSE's access to the flash memory. For details, refer "Flash Block Assignment" spreadsheet attached with the document.

82.1.1.1 CSE SECURE_BOOT execution, and other configuration

The DCF client configures various aspects of CSE operation. For details about Secure boot configurations, refer DCF_CSE_SECURE_BOOT and DCF_CSE_SECURE_BOOT_SIZE in 'UTEST DCF clients' and 'DCF Client Register Bits' tabs of the DCF sheet spreadsheet attached with this document. The settings in the DCF client load during reset.

For details about how and when the SECURE_BOOT command is issued in this chip, refer SECURE BOOT details.

82.1.1.2 Making secure flash blocks accessible for the Failure Analysis life cycle

Before advancing the chip's life cycle to Failure Analysis, make its secure flash blocks accessible. To do so, issue the DEBUG_CHAL and DEBUG_AUTH commands described in Debug Challenge and Debug Authorization.

82.1.1.3 CSE operation while JTAGM is in use

In order to use CSE memory slots for decryption/encryption which have the DEBUG_PROT flag set, the user must disable the JTAGM module by writing JTAGM_MCR[DTM] = 0b0 and give short functional reset to clear CSE_SR[EDB].

The JTAGM module is disabled by default, so if it is not used in the application then no action is necessary.

82.1.1.4 CSE command-processing completion before Stop mode entry

To ensure proper CSE operation: Before the chip enters Stop mode, CSE_SR[BSY] must be 0.

82.2 Introduction

82.2.1 Overview

The Cryptographic Services Engine (CSE) is a peripheral module that implements the security functions described in the *Secure Hardware Extension (SHE) Functional Specification Version 1.1*.

The CSE design includes a host interface with a set of memory mapped registers that are used by the CPU to issue commands and a system bus interface that allows the CSE to directly access system memory. Two dedicated blocks of system flash memory are used by the CSE for secure key storage.

82.2.2 Features

The CSE has the following features:

- Secure storage for cryptographic keys
- AES-128 encryption and decryption
- AES-128 CMAC authentication
- True random number generation
- Secure boot mode
- System bus master interface

82.2.3 Modes of operation

The CSE supports operation in Normal and Debug modes of operation. The use of the cryptographic keys stored by the CSE is controlled based on the activation of the CPU debug port and the successful completion of the secure boot process.

The CSE has a low-power mode that disables the clock to all logic except the host interface. Register accesses are supported in this mode, but commands are not processed.

82.2.4 Block diagram

The CSE design includes a command processor, host interface, system bus interface, local memory, AES logic, and True Random Number Generator (TRNG) as shown below.

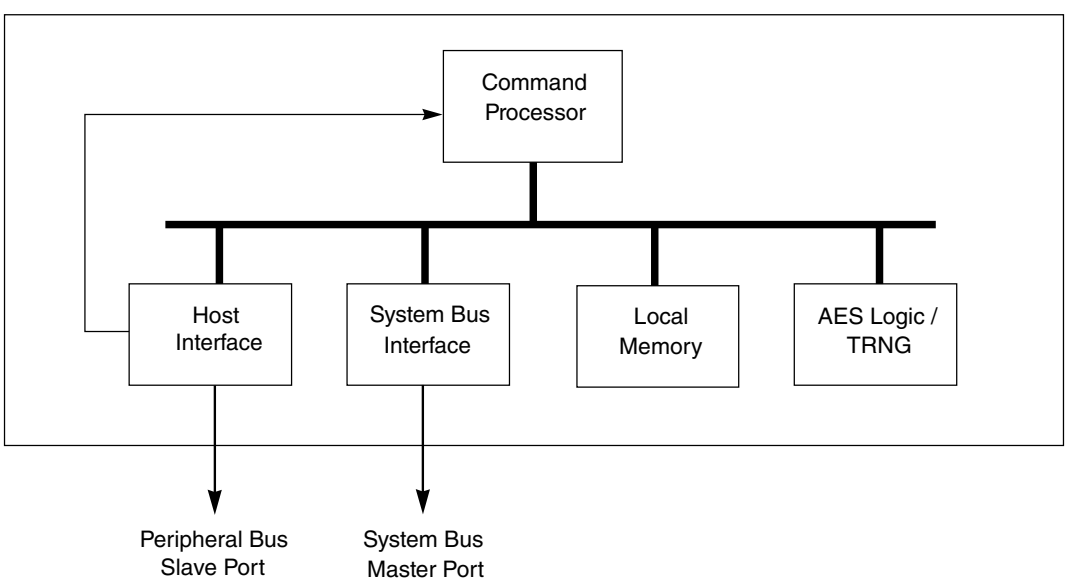


Figure 82-1. CSE block diagram

82.3 External signal description

The CSE has no external interface signals.

82.4 Memory Map and Register Definition

The CSE programming model has ten 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. The CSE memory map is shown in the table below.

CSE memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CSE Control Register (CSE_CR)	32	R/W	0000_0000h	82.4.1/4030
4	CSE Status Register (CSE_SR)	32	R	0000_0000h	82.4.2/4032
8	CSE Interrupt Register (CSE_IR)	32	w1c	0000_0000h	82.4.3/4033
C	CSE Error Code Register (CSE_ECR)	32	R	0000_0000h	82.4.4/4034
20	CSE Command Register (CSE_CMD)	32	R/W	0000_0000h	82.4.5/4035
24	CSE Parameter Register (CSE_P1)	32	R/W	0000_0000h	82.4.6/4036
28	CSE Parameter Register (CSE_P2)	32	R/W	0000_0000h	82.4.6/4036
2C	CSE Parameter Register (CSE_P3)	32	R/W	0000_0000h	82.4.6/4036
30	CSE Parameter Register (CSE_P4)	32	R/W	0000_0000h	82.4.6/4036
34	CSE Parameter Register (CSE_P5)	32	R/W	0000_0000h	82.4.6/4036

82.4.1 CSE Control Register (CSE_CR)

The CSE_CR contains fields for configuring and controlling operation of the CSE.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIV								0		KBS	SFE	MDIS	SUS	DRE	CIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 DIV	TRNG Clock Divider Select. Sets the clock divide ratio for the TRNG clock. The TRNG clock is the module clock divided by a programmable ratio: TRNG Clk Freq = CSE Clk Freq / (2*(DIV+1)) The divide ratio must be set such that the TRNG clock frequency is between 500 kHz and 2MHz. 0x00 divide by 2 0x01 divide by 4 0x02 divide by 6 ... 0xFE divide by 510 0xFF divide by 512
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 KBS	Key Bank Select. Selects which set of application keys are visible for commands. See Table 82-1. 0 Select KEY_1-KEY_10 1 Select KEY_11-KEY_20
27 SFE	Security Flag Extension. When the SFE bit is set, the CSE recognizes the VERIFY_ONLY security flag during LOAD_KEY processing; otherwise, it is ignored. The SFE bit has no effect on the enforcement of the VERIFY_ONLY security flag once a key is loaded. 0 Security Flag Extension disabled. 1 Security Flag Extension enabled.
28 MDIS	Module Disable. When the MDIS bit is set, the CSE is put into a low-power mode that disables the clock to all of the CSE logic except for the host interface. The MDIS bit should not be set during command processing (CSE_SR[BSY] = 1). The current command should be canceled and processing stopped (CSE_SR[BSY] = 0) before setting the MDIS bit. 0 Normal mode 1 Low-power mode
29 SUS	Suspend Command Processing. When the SUS bit is set, the CSE suspends processing of the current command until the SUS bit is cleared. The current execution status of the command processor is reflected by the CSE_SR[EX] flag. 0 Enable processing of commands. 1 Suspend command processing.
30 DRE	DMA Request Enable. When the DRE bit is set, the DMA request is asserted when command processing is completed. 0 DMA request not enabled. 1 DMA request enabled.
31 CIE	Command Complete Interrupt Enable. When the CIE bit is set, an interrupt request is generated when command processing is completed. 0 Command Complete Interrupt disabled. 1 Command Complete Interrupt enabled.

82.4.2 CSE Status Register (CSE_SR)

The CSE_SR contains flags indicating the status of the CSE. Reading CSE_SR[24:31] is the same as the SHE GET_STATUS command.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							EX	IDB	EDB	RIN	BOK	BFN	BIN	SB	BSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_SR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 EX	Execute. **NOTE:** Ignore the status of CSE_SR[EX] when CSE_SR[BSY] flag is '0'. 0 Command processor is idle or suspended. 1 Command processor is executing a command.
24 IDB	Internal Debug. The IDB bit is set by the SECURE_BOOT, INIT_CSE, and DEBUG_AUTH commands when no user keys (i.e. all keys except SECRET_KEY and UID) are stored. It is cleared on reset and by the LOAD_KEY command. 0 Internal Debug functions disabled. 1 Internal Debug functions enabled.
25 EDB	External Debug. The EDB bit is set when the CPU debug port is activated and is cleared on reset. 0 External debugger not attached. 1 External debugger attached.
26 RIN	Random Number Generator Initialized. The RIN bit is set by the INIT_RNG command and is cleared on reset and by the DEBUG_AUTH command. 0 Random number generator not initialized. 1 Random number generator initialized.
27 BOK	Secure Boot OK. The BOK bit is set by the successful completion of the SECURE_BOOT command. It is cleared by reset and by the BOOT_FAILURE command.

Table continues on the next page...

CSE_SR field descriptions (continued)

Field	Description
	0 Secure boot not completed or secure boot failure. 1 Secure boot successful.
28 BFN	Secure Boot Finished. The BFN bit is set by the SECURE_BOOT command when the BIN bit is set, when an error is encountered, or when the BOOT_MAC value does not match. It is also set by the BOOT_OK or BOOT_FAILURE commands. It is cleared on reset. The BFN bit is not set in the event the CSE flash blocks are accessible, but contain an invalid firmware image as detected by a checksum mismatch (CSR_ECR=0x13). 0 Secure boot not finished. 1 Secure boot finished.
29 BIN	Secure Boot Initialization. The BIN bit is set by the SECURE_BOOT command if the BOOT_MAC memory slot is empty and is cleared on reset. 0 Secure boot personalization not completed. 1 Secure boot personalization completed.
30 SB	Secure Boot. The SB bit is set by the SECURE _BOOT command if the BOOT_MAC_KEY slot is not empty, and is cleared on reset. 0 Secure boot not activated. 1 Secure boot activated.
31 BSY	Busy. The BSY bit is set when a command is issued and cleared when command processing is completed. 0 Command processing completed. 1 Command processing not completed.

82.4.3 CSE Interrupt Register (CSE_IR)

The CSE_IR contains the command completion interrupt flag bit.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Command Complete Interrupt Flag. The CIF flag asserts after a command is completed. The CIF flag and the associated interrupt request are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 Command not complete. 1 Command completed.

82.4.4 CSE Error Code Register (CSE_ECR)

The CSE_ECR contains the error code from the last completed command.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											EC				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_ECR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EC	Error Code from the last command completed. 0x00 No error 0x02 Command sequence error 0x03 Key not available 0x04 Invalid key 0x05 Empty key 0x06 No secure boot 0x07 Key write protected 0x08 Key update error 0x09 Random number seed not initialized 0x0A Internal debug not allowed 0x0B Command issued while busy 0x0C System memory error 0x10 Internal memory error 0x11 Invalid command

Table continues on the next page...

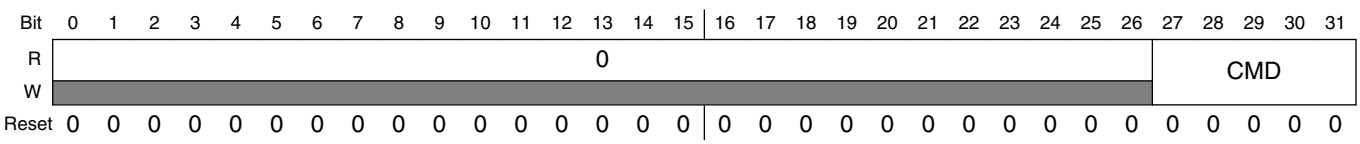
CSE_ECR field descriptions (continued)

Field	Description
	0x12 TRNG error 0x13 CSE flash block error 0x14 Internal command processor error 0x15 Length error

82.4.5 CSE Command Register (CSE_CMD)

Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD). Reads of the CSE_CMD register have no effect on the operation of the CSE. See Command processing for a description of command processing and CSE commands for a description of each command.

Address: 0h base + 20h offset = 20h



CSE_CMD field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CMD	Command. See CSE commands for details of each command. 0x01 = ENC_ECB 0x02 = ENC_CBC 0x03 = DEC_ECB 0x04 = DEC_CBC 0x05 = GENERATE_MAC 0x06 = VERIFY_MAC 0x07 = LOAD_KEY 0x08 = LOAD_PLAIN_KEY 0x09 = EXPORT_RAM_KEY 0x0A = INIT_RNG 0x0B = EXTEND_SEED 0x0C = RND

Table continues on the next page...

CSE_CMD field descriptions (continued)

Field	Description
	0x0D = SECURE_BOOT
	0x0E = BOOT_FAILURE
	0x0F = BOOT_OK
	0x10 = GET_ID
	0x11 = CANCEL
	0x12 = DEBUG_CHAL
	0x13 = DEBUG_AUTH
	0x14 = TRNG_RND
	0x15 = INIT_CSE
	0x16 = MP_COMPRESS

82.4.6 CSE Parameter Register (CSE_P*n*)

The five parameter registers (CSE_Px) are used to hold command parameter values. See Command processing for a description of how to issue commands and CSE commands for a description of each command. The parameter registers are read only when the CSE_SR[BSY] bit is set.

Address: 0h base + 24h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PARM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_P*n* field descriptions

Field	Description
0–31 PARM	Command parameter (data value or address of data value).

82.5 CSE functional description

The CSE implements a comprehensive set of cryptographic functions as described in the *SHE Functional Specification*, including secure key storage, AES encryption, secure boot, AES CMAC authentication, and random number generation.

82.5.1 Host interface

The host interface includes all of the CSE memory mapped registers (see the memory map and register section).

The Control Register (CSE_CR) is used to set configuration options and control operation of the CSE. If the CSE_CR[CIE] bit is set, an interrupt request is generated after the CSE finishes processing each command. The interrupt is cleared by writing a 1 to the CSE_IR[CIF] bit. If present, the CSE also supports servicing by the system DMA module. The DMA request is asserted after the CSE finishes processing each command if the CSE_CR[DRE] bit is set. When the CSE_CR[SUS] bit is set, the CSE suspends processing of the current command and does not respond to new commands including the CANCEL command.

Command processing is resumed by clearing the CSE_CR[SUS] bit. The CSE cannot respond immediately to changes in the CSE_CR[SUS] bit, so the CSE_SR[EX] flag indicates the current status of the CSE command processor. The CSE_SR[EX] flag is set when the CSE command processor is running and is cleared when the CSE is idle or has suspended processing. The CSE_CR[MDIS] bit puts the CSE into a low-power mode. In low-power mode, registers can be accessed but commands are not processed. Before setting the CSE_CR[MDIS] bit or entering any device low-power modes, application software should cancel any pending command and wait for the CSE_SR[BSY] bit to be cleared. The CSE_CR[SFE] bit enables loading of the VERIFY_ONLY flag during the LOAD_KEY command. The CSE_CR[KBS] bit selects which key bank (KEY_1–KEY10 or KEY_11–KEY_20) is active.

The Status Register (CSE_SR) contains a set of nine status flags. Reading CSE_SR[24:31] is equivalent to the SHE GET_STATUS command. The CSE_SR[BSY] bit indicates when the CSE is processing a command. The flag is cleared when processing is completed. Polling the CSE_SR[BSY] bit is an alternative to interrupt-driven operation. The CSE_SR[SB], CSE_SR[BIN], CSE_SR[BFN] and CSE_SR[BOK] bits reflect the secure boot status (see Secure boot). The CSE_SR[RIN] bit is set when the PRNG is initialized with a seed value generated by the TRNG. The CSE_SR[EDB] is set if an external debugger is connected. The CSE_SR[IDB] flag is set when internal debugging has been enabled with the DEBUG_CHAL and DEBUG_AUTH commands. The CSE_SR is a read-only register and can be read at any time.

The Error Code Register (CSE_ECR) contains the error code from the last completed command. The CSE_ECR is set to 0 if no error occurred while processing the command. The CSE_ECR is a read-only register.

82.5.2 Command processing

CSE functions are accessed through a set of commands that are listed in CSE commands. Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD). CSE_CMD register reads have no effect on the operation of the CSE.

Command inputs and outputs that are 32 bits or less in size are stored directly into a parameter register. Inputs and outputs that are larger than 32 bits are kept in system memory and the address of the data (pointer) is loaded into the parameter register. All data must be aligned on a 32-bit (word) boundary. The two least significant bits of an address are ignored. All data is stored in big-endian format with the most significant byte of the data block stored in the lowest address.

The first command issued to the CSE after reset must be either SECURE_BOOT or INIT_CSE. In some device boot modes, system boot logic or BAM code will issue the SECURE_BOOT command, which initializes the CSE and starts the SHE secure boot protocol (see Secure boot). For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information. For the other boot modes, application software must issue the INIT_CSE command before issuing other CSE commands. The INIT_CSE command can only be issued one time after reset and only in boot modes that do not issue the SECURE_BOOT command. Application software cannot issue the SECURE_BOOT command.

The CSE can only process one command at a time. Commands take a number of clock cycles to complete, so one of the following must to be used to determine when the output data is valid and the next command can be issued:

- The CSE_SR[BSY] flag
- The interrupt request
- The DMA request

Parameter registers cannot be modified while the CSE is processing a command. Command processing can be suspended and resumed using the CSE_CR[SUS] bit (see Host interface). If the CSE_CMD register is written while a command is being processed, processing is aborted and an error is generated with CSE_ECR[EC] = 0x0B. The CANCEL command can be used to abort processing of a command without generating an error. The CSE_ECR is updated and the CSE_SR[BSY] bit is cleared upon completion of command processing (see Error handling).

82.5.3 Secure storage

The CSE provides secure, nonvolatile storage for cryptographic keys as described in the *SHE Functional Specification*. The keys are stored in 25 memory slots, with one ROM slot, 23 nonvolatile slots, and one RAM slot as shown in Table 82-1. The first four slots have a dedicated use; the other slots are available for application specific keys. The CSE extends the SHE specification by providing 10 extra application key slots (KEY_11–KEY_20) which are enabled by setting the CSE_CR[KBS] bit. The BOOT_MAC slot is loaded with a MAC value used by the secure boot process. All other slots are used for encryption or message authentication keys. The SECRET_KEY slot is programmed with a random value during device fabrication. All CSE encryption and message authentication commands specify a key by its Key ID.

Table 82-1. Memory slots

Slot name	KBS	Key ID	Type
SECRET_KEY	x	0x0	ROM
MASTER_ECU_KEY	x	0x1	nonvolatile
BOOT_MAC_KEY	x	0x2	nonvolatile
BOOT_MAC	x	0x3	nonvolatile
KEY_1–KEY_10	0	0x4–0xD	nonvolatile
KEY_11–KEY_20	1	0x4–0xD	nonvolatile
RAM_KEY	x	0xE	RAM

In addition to the 25 memory slots, the CSE holds a 120-bit read-only unique identification number (UID) that is programmed during device fabrication. The UID is used in the memory update procedure and is available for application-specific uses.

Each memory slot holds a 128-bit value, 6 security flags, and a 28-bit counter. See the *SHE Functional Specification* for the required structure needed for the LOAD_KEY command. The security flags are defined in Table 82-2. The 28-bit counter must be advanced each time a new key is loaded. The counter value is zero for an empty memory slot.

Table 82-2. Memory slot security flags

Flag name	Description
VERIFY_ONLY	If set, the memory slot key cannot be used by the GENERATE_MAC command, only by the VERIFY_MAC command. This bit has no effect if KEY_USAGE = 0.
WRITE_PROT	If set, the memory slot cannot be updated.
BOOT_PROT	If set, the memory slot is disabled if CSE_SR[BOK] = 0 or CSE_SR[BFN] = 0.
DEBUG_PROT	If set, the memory slot is disabled if CSE_SR[EDB] = 1.
KEY_USAGE	If set, the memory slot holds a MAC key; otherwise, it holds an encryption key.
WILDCARD	If set, the memory slot cannot be updated with the wildcard UID.

The memory slot security flags for a specified key are checked against the associated bits in the CSE_SR during command processing as described in the SHE specification. If the memory slot is disabled due to security flag settings, a key not available error (EC = 0x03) is returned by the command. The VERIFY_ONLY security flag is a CSE-specific feature that is not defined in the SHE specification. If the CSE_CR[SFE] bit is set when the LOAD_KEY command is executed, the security flag field (FID) in the M2 message is extended by one bit with the VERIFY_ONLY flag and the adjacent zero fill field is reduced to 94 bits. The CSE_CR[SFE] bit has no effect on the enforcement of the VERIFY_ONLY flag for commands issued after the LOAD_KEY.

A key is loaded into a memory slot using the LOAD_KEY command which implements the SHE memory update protocol. An empty key slot can only be used as the authorization key to update itself (KeyID = AuthID). In this case, the authorization key has a value of 0. The BOOT_PROT and DEBUG_PROT security flags are not enforced for the LOAD_KEY command except when the RAM_KEY slot is loaded. In that case, the BOOT_PROT and DEBUG_PROT flags of the AuthID slot are applied. Additionally, when loading the RAM_KEY slot, the counter and security flag fields in the M2 message must be zero.

The LOAD_PLAIN_KEY command can be used to load a key into the RAM_KEY slot. The EXPORT_RAM_KEY command can then be used to export the key in an encrypted form compatible with the LOAD_KEY command for storage outside the CSE. The BOOT_PROT and DEBUG_PROT security flags of the MASTER_ECU_KEY are enforced for the EXPORT_RAM_KEY command.

Two dedicated blocks in the system flash memory are used to implement the secure storage feature. These blocks are not program visible and only the CSE can read, erase, and program these blocks. The CSE reads and writes to the system flash memory via the system bus master interface during the SECURE_BOOT, INIT_CSE, DEBUG_AUTH, and LOAD_KEY commands. During execution of these commands, the system MPU (if present) must be configured to allow the CSE access to the system flash memory and other bus masters must be programmed properly to avoid interfering with the CSE.

82.5.4 Encryption and decryption

The CSE supports AES-128 encryption and decryption in ECB and CBC modes of operation. The key is selected from one of the memory slots, which must be enabled for the operation. A plaintext key can be loaded into the RAM_KEY slot using the LOAD_PLAIN_KEY command for keys that are not stored in a nonvolatile memory slot.

82.5.5 Message authentication

The CSE uses the AES-128 CMAC algorithm for message authentication. The key for the CMAC operation is selected from one of the memory slots which must be enabled for the operation. A plaintext key can be loaded into the RAM_KEY slot using the LOAD_PLAIN_KEY command for keys that are not stored in a nonvolatile memory slot. The VERIFY_MAC command supports comparison of a calculated MAC with an input MAC value.

82.5.6 Secure boot

The CSE implements the SHE secure boot protocol. In some device boot modes, the BAM or system boot logic issues the SECURE_BOOT command to the CSE, which starts the secure boot process. For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information.

The first step in the process is for the CSE to download the command processor firmware and memory slot data from the CSE flash memory blocks into local memory. If the BOOT_MAC_KEY slot is empty, the CSE_SR[SB] flag is cleared and the process is finished. Otherwise, the CSE_SR[SB] flag is set and CSE calculates the MAC over the specified bootloader code. If the BOOT_MAC slot is empty, the calculated MAC is loaded into the BOOT_MAC slot, the CSE_SR[BIN] and CSE_SR[BFN] flags are set, and the process is finished. Otherwise, the calculated MAC value is compared to the value in the BOOT_MAC slot. If the values match, the CSE_SR[BOK] bit is set; otherwise, the CSE_SR[BFN] bit is set. If the CSE_SR[BOK] flag is set, the user boot code can issue the BOOK_OK command which sets the CSE_SR[BFN] bit. The memory slots that have the BOOT_PROT flag set are enabled when both the CSE_SR[BFN] and CSE_SR[BOK] flags are set.

The SECURE_BOOT command can run either before the user boot code is executed or in parallel with the user boot code. If the parallel mode is used, the user boot code must suspend command processing (set CSE_CR[SUS] to 1) while performing any operations —such as configuring the flash controller—that may interfere with the CSE accessing flash memory. A device-specific bit selects whether to run the SECURE_BOOT command before or in parallel with the user boot code. For details about this bit, see the chip-specific CSE information.

The CSE implements an internal timer that is started when the SECURE_BOOT command is issued. The initial value of the timer is set to 0x00F4_2400 (16,000,000 CSE system clocks). If the timer expires before the SECURE_BOOT command completes, any application code intended to be blocked by the SECURE_BOOT process will instead see the non-completed SECURE_BOOT status of CSE_SR[BSY]=1 and

CSE_SR[BOK]=0. In the event of a SECURE_BOOT timeout, the CSE will continue executing the SECURE_BOOT command until completion, until suspended, or until receiving the CANCEL command from the application code. There is no error code associated with a SECURE_BOOT timeout.

82.5.7 Random number generation

The CSE has both a Pseudo Random Number Generator (PRNG) and a True Random Number Generator (TRNG).

The PRNG has a 128-bit state variable and uses AES in output feedback mode to generate pseudo random values. A key derived from the SECRET_KEY is used for the PRNG. The RND command updates the state of the PRNG and returns the 128-bit random value. The EXTEND_SEED command can be used to add entropy to the PRNG state. The PRNG state is initialized after each reset with the INIT_RNG command, which uses the TRNG to generate a 128-bit seed value for the PRNG. The CSE_SR[RIN] flag is set when the PRNG is initialized.

The INIT_RNG and TRNG_RND commands use the TRNG to generate truly random values. The TRNG hardware runs off of a slower clock derived from the system clock. The CSE_CR[DIV] field needs to be configured for these commands such that the TRNG clock is between 500 kHz and 2 MHz. Random values generated by the TRNG are checked with a statistical test to verify proper operation of the TRNG. If the test fails, a TRNG error (EC = 0x12) is returned. Due to the statistical nature of this test, there is a very small probability ($< 10^{-9}$) that a properly operating TRNG will return an error. If an TRNG error is returned, the command can be issued again.

82.5.8 Error handling

When the CSE command processor encounters an error condition, it stops processing and returns an error code as described in Table 82-3. The CSE_SR[BSY] and CSE_SR[EX] bits are cleared with interrupt and/or DMA requests generated the same as if the command had completed successfully. In most cases, error conditions are detected before data processing begins and no output values are written. Intermediate or invalid output data is never written to the parameter registers or system memory. However, it is possible for outputs to system memory to be partially written when an error occurs. The CSE does not zero out this partially written data.

Table 82-3. Error code summary

Error code	Error description	Error conditions
0x00	No error	1. Command successfully executed with no errors encountered.
0x02	Command sequence error	1. Command issued (except for CANCEL) before the SECURE_BOOT or INIT_CSE command. 2. The DEBUG_AUTH command issued before the DEBUG_CHAL command. 3. SECURE_BOOT or INIT_CSE command issued after an initial SECURE_BOOT or INIT_CSE is issued.
0x03	Key not available	1. A required key is not available due to a BOOT_PROT or DEBUG_PROT security flag restriction (see Secure storage).
0x04	Invalid key	1. The specified key slot is not valid for the command. (See the SHE specification.) 2. The specified key slot is not available due to a KEY_USAGE security flag restriction (see Secure storage).
0x05	Empty key	1. The specified key slot is empty.
0x06	No secure boot	1. SECURE_BOOT issued when secure boot is disabled or the BOOT_MAC_KEY slot is empty. 2. BOOT_FAIL or BOOT_OK issued with incorrect settings for either the CSE_SR[SB], CSE_SR[BFN] or CSE_SR[BOK] flags.
0x07	Key write protected	1. Attempt to load a key slot with the WRITE_PROT security flag set. 2. Attempt to enter internal debug mode (DEBUG_CHAL, DEBUG_AUTH) with a WRITE_PROT security flag set in one or more key slots.
0x08	Key update error	The LOAD_KEY command failed due to one of the following conditions: 1. The M3 MAC value is invalid. 2. The wildcard UID is specified with the WILDCARD flag set. 3. The specified UID does not match the device UID. 4. The update counter or security flag values are not zero when loading the RAM_KEY slot. 5. The specified update counter value is not greater than the current counter value for the slot. (Counter value is zero for an empty slot.)
0x09	Random number seed not initialized	1. RND, EXTEND_SEED, or DEBUG_CHAL command issued before the INIT_RNG command.
0x0A	Internal debug not allowed	1. DEBUG_AUTH command issued with invalid MAC value.
0x0B	Command issued while busy	1. Command issued when the CSE_SR[BSY] bit is set.
0x0C	System memory error	1. A system memory error was encountered while executing the command. (The CSE flash memory block error code is generated for bus errors encountered when accessing the CSE flash memory blocks.)
0x10	Internal memory error	1. An internal memory error was encountered while executing the command.

Table continues on the next page...

Table 82-3. Error code summary (continued)

Error code	Error description	Error conditions
0x11	Invalid command	1. Value written to CSE_CMD register is out of range.
0x12	TRNG error	1. One or more statistical tests run on the TRNG output failed (see Generate Random Number).
0x13	CSE flash memory block error	1. Error reading, programming, or erasing one of the CSE flash memory blocks. 2. UID or SECRET_KEY required but not available.
0x14	Internal command processor error	1. An internal error condition was encountered while executing the command.
0x15	Length error	1. MAC length for VERIFY_MAC command is greater than 128. 2. Message length for GENERATE_MAC, VERIFY_MAC, or MP_COMPRESS command is greater than 0x7FFFFFFF (4 GB).

82.6 CSE commands

This section describes the set of CSE commands. Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD) as described in Command processing. The first command issued to the CSE after reset must be INIT_CSE for device boot modes that do not support secure boot. Command inputs and outputs that are 32 bits or less in size are stored directly in a parameter register. Inputs and outputs that are larger than 32 bits are kept in system memory and the address of the data (pointer) is loaded into the parameter register. The data direction indication in the tables below refers to the command parameter value which may be stored in memory.

82.6.1 Encrypt ECB

The ENC_ECB command performs AES-128 encryption in ECB mode on n 128-bit blocks of data with the parameters specified in the table below.

Table 82-4. ENC_ECB command

Register	Value	Data direction
CSE_CMD	0x01	—
CSE_P1	Key ID	Input
CSE_P2	Number of blocks (*n*)	Input

Table continues on the next page...

Table 82-4. ENC_ECB command (continued)

Register	Value	Data direction
CSE_P3	First plaintext block address	Input
CSE_P4	First ciphertext block address	Output

82.6.2 Encrypt CBC

The ENC_CBC command performs AES-128 encryption on n 128-bit blocks of data in CBC mode with the parameters specified in the table below. The number of blocks parameter is a 32-bit value.

Table 82-5. ENC_CBC command

Register	Value	Data direction
CSE_CMD	0x02	—
CSE_P1	Key ID	Input
CSE_P2	IV address	Input
CSE_P3	Number of blocks (*n*)	Input
CSE_P4	First plaintext block address	Input
CSE_P5	First ciphertext block address	Output

82.6.3 Decrypt ECB

The DEC_ECB command performs AES-128 ECB decryption on *n* 128-bit blocks of data with the parameters specified in the table below.

Table 82-6. DEC_ECB command

Register	Value	Data direction
CSE_CMD	0x03	—
CSE_P1	Key ID	Input
CSE_P2	Number of blocks (*n*)	Input
CSE_P3	First ciphertext block address	Input
CSE_P4	First plaintext block address	Output

82.6.4 Decrypt CBC

The DEC_CBC command performs AES-128 decryption in CBC mode on *n* 128-bit blocks of data with the parameters specified in the table below. The number of blocks parameter is a 32-bit value.

Table 82-7. DEC_CBC command

Register	Value	Data direction
CSE_CMD	0x04	—
CSE_P1	Key ID	Input
CSE_P2	IV address	Input
CSE_P3	Number of blocks (*n*)	Input
CSE_P4	First ciphertext block address	Input
CSE_P5	First plaintext block address	Output

82.6.5 Generate MAC

The GENERATE_MAC command calculates the MAC of a given message with the parameters specified in the table below. The AES CMAC algorithm is used to calculate a 128-bit MAC output. The message length input is a 64-bit value that specifies the length of the message in bits. A length error (EC = 0x15) is returned if the message length is greater than 0x7fffffff (4 GB).

Table 82-8. GENERATE_MAC command

Register	Value	Data direction
CSE_CMD	0x05	—
CSE_P1	Key ID	Input
CSE_P2	Message length (bits) address	Input
CSE_P3	Message start address	Input
CSE_P4	MAC address	Output

82.6.6 Verify MAC

The VERIFY_MAC command verifies the MAC of a given message with the parameters specified in Table 82-9. The AES CMAC algorithm is used to calculate a 128-bit MAC that is truncated according to the MAC length parameter, which specifies the number of most significant bits in the MAC to compare. A MAC length value of 0 indicates that all 128 bits are compared; a value greater than 128 returns a length error (EC = 0x15). The

message length input is a 64-bit value that specifies the length of the message in bits. A length error (EC = 0x15) is returned if the message length is greater than 0x7fffffff (4 GB). If the input MAC matches the MAC calculated over the message, the CSE_P5 register is set to 0; otherwise, it is set to 1.

Table 82-9. VERIFY_MAC command

Register	Value	Data direction
CSE_CMD	0x06	—
CSE_P1	Key ID	Input
CSE_P2	Message length (bits) address	Input
CSE_P3	Message start address	Input
CSE_P4	MAC address	Input
CSE_P5	MAC length (bits)/ Verification status	Input / Output

82.6.7 Load Key

The LOAD_KEY command updates a memory slot using parameters specified in Table 82-10 according to the SHE memory slot update protocol (see Secure storage). The 128-bit M1 message contains the UID, Key ID, and Authentication Key ID. The 256-bit M2 message contains the new security flags, the counter, and the key value all encrypted using a derived key generated from the Authentication Key. The 128-bit M3 message is a MAC generated over messages M1 and M2. The 256-bit M4 message is the concatenation of the UID, Key ID, Authorization Key ID, and the encrypted counter value. The 128-bit M5 message is the MAC calculated over message M4.

The derived keys used to generate messages M2, M3, M4, and M5 deviate from the SHE memory slot update protocol when updating the second bank of user keys (KEY_11–KEY_20). During the generation of M2 and M4, K1 and K3 shall be derived using a constant equal to KEY_UPDATE_ENC_C + 0x00800000_00000000_00000000_00000000. Similarly, during generation of M3 and M5, K2 and K4 shall be derived using a constant equal to KEY_UPDATE_MAC_C + 0x00800000_00000000_00000000_00000000. This is intended to prevent aliasing between the two banks of user keys.

Table 82-10. LOAD_KEY command

Register	Value	Data direction
CSE_CMD	0x07	—
CSE_P1	M1 address	Input

Table continues on the next page...

Table 82-10. LOAD_KEY command (continued)

Register	Value	Data direction
CSE_P2	M2 address	Input
CSE_P3	M3 address	Input
CSE_P4	M4 address	Output
CSE_P5	M5 address	Output

82.6.8 Load Plain Key

The LOAD_PLAIN_KEY command updates the RAM key memory slot with a 128-bit plaintext key with the parameter shown in the table below.

Table 82-11. LOAD_PLAIN_KEY command

Register	Value	Data direction
CSE_CMD	0x08	—
CSE_P1	Key address	Input

82.6.9 Export RAM Key

The EXPORT_RAM_KEY command exports the RAM key data with the parameters in the table below. Only keys loaded with the LOAD_PLAIN_KEY command may be exported. The output messages are compatible with the messages used for LOAD_KEY (see Load Key for details).

Table 82-12. EXPORT_RAM_KEY command

Register	Value	Data direction
CSE_CMD	0x09	—
CSE_P1	M1 address	Output
CSE_P2	M2 address	Output
CSE_P3	M3 address	Output
CSE_P4	M4 address	Output
CSE_P5	M5 address	Output

82.6.10 Initialize RNG

The INIT_RNG command initializes the internal PRNG state with a seed value generated by the TRNG and sets the CSE_SR[RIN] flag. It takes 2048 TRNG clock cycles to generate a seed value. The CSE_CR[DIV] field must be properly configured before this command is executed (see register descriptions for further information). There is a very small probability that this command may return a TRNG error (EC = 0x12) even when the TRNG is operating properly (see Random number generation). The INIT_RNG command must be called after each reset before the RND command is issued. The command has no parameters as shown the table below.

Table 82-13. INIT_RNG command

Register	Value	Data direction
CSE_CMD	0x0A	—

82.6.11 Extend PRNG Seed

The EXTEND_SEED command extends the state of the PRNG using a 128-bit entropy input value. The current PRNG state and the input data are compressed into a new PRNG state value. This command is issued with the parameter shown in the table below.

Table 82-14. EXTEND_SEED command

Register	Value	Data direction
CSE_CMD	0x0B	—
CSE_P1	Entropy value address	Input

82.6.12 Generate Random Number

The RND command generates a 128-bit random value and updates the state of the internal PRNG. The PRNG state must be initialized after reset using the INIT_RNG command before this command can be issued. This command is issued with the parameter shown in the table below

Table 82-15. RND command

Register	Value	Data direction
CSE_CMD	0x0C	—
CSE_P1	Random value address	Output

82.6.13 Secure Boot

The SECURE_BOOT command loads the command processor firmware and memory slot data from the CSE flash memory blocks, and then it executes the SHE secure boot protocol using the parameters shown in Table 82-16 (see Secure boot for details). The bootloader size is a 32-bit value. The CSE firmware version is loaded into CSE_P3 register.

The SECURE_BOOT command is issued during the boot processes by the system boot logic. For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information.

Table 82-16. SECURE_BOOT command

Register	Value	Data direction
CSE_CMD	0x0D	—
CSE_P1	Bootloader size (bytes)	Input
CSE_P2	Bootloader start address	Input
CSE_P3	Firmware version	Output

82.6.14 Boot Failure

The BOOT_FAILURE command sets the CSE_SR[BFN] flag and clears the CSE_SR[BOK] flag to disable use of memory slots that have the BOOT_PROT flag set. This command is issued with no parameters as shown in the table below.

Table 82-17. BOOT_FAILURE command

Register	Value	Data direction
CSE_CMD	0x0E	—

82.6.15 Boot OK

The BOOT_OK command sets the CSE_SR[BFN] flag and leaves the CSE_SR[BOK] flag set to confirm successful completion of the secure boot processes. This enables the use of memory slots that have the BOOT_PROT flag set if they are not disabled for some other reason. This command is issued with no parameters as shown in the table below.

Table 82-18. BOOT_OK command

Register	Value	Data direction
CSE_CMD	0x0F	—

82.6.16 Get ID

The GET_ID command returns the UID, CSE_SR[24:31], and a 128-bit MAC calculated over the concatenation of a 128-bit input challenge value, UID, and CSE_SR[24:31]. The MASTER_ECU_KEY is used for the MAC calculation. A value of 0 is returned for the MAC if the MASTER_ECU_KEY slot is empty. The UID output is a 128-bit value with the 8 least significant bits set to 0. This command is issued with the parameters shown in the table below.

Table 82-19. GET_ID command

Register	Value	Data direction
CSE_CMD	0x10	—
CSE_P1	Challenge address	Input
CSE_P2	UID output address	Output
CSE_P3	CSE_SR[24:31] value	Output
CSE_P4	MAC address	Output

82.6.17 Cancel

The CANCEL command aborts processing of the current command and clears the CSE_SR[BSY] flag. This command is issued with no parameters as shown in the table below.

Table 82-20. CANCEL command

Register	Value	Data direction
CSE_CMD	0x11	—

82.6.18 Debug Challenge

The DEBUG_CHAL command generates a 128-bit random challenge output value that is used in conjunction with the DEBUG_AUTH command. The PRNG state must be initialized after reset using the INIT_RNG command before this command can be issued. This command is issued with the parameter shown in the table below.

Table 82-21. DEBUG_CHAL command

Register	Value	Data direction
CSE_CMD	0x12	—
CSE_P1	Challenge address	Output

82.6.19 Debug Authorization

The DEBUG_AUTH command erases all user keys and sets the CSE_SR[IDB] flag which enables internal debugging if the 128-bit authorization input value is valid and no memory slots are write protected. The authorization input is generated using the DEBUG_CHAL command output and the UID as described in the *SHE Functional Specification*. If the DEBUG_CHAL command is not issued before the DEBUG_AUTH command, a command sequence error (EC = 0x02) is returned. However, other commands may be issued between the DEBUG_CHAL and DEBUG_AUTH commands. This command is issued with the parameter shown in the table below.

Table 82-22. DEBUG_AUTH command

Register	Value	Data direction
CSE_CMD	0x13	—
CSE_P1	Authorization address	Input

82.6.20 Generate TRNG Random Number

The TRNG_RND command generates a 128-bit random output value using the TRNG. It takes 2048 TRNG clock cycles to generate a random value. The CSE_CR[DIV] field must be properly configured before this command is executed (see the register descriptions section for further information). This command takes much longer to execute than the RND command that should normally be used to generate random values. There is a very small probability that this command may return a TRNG error (EC = 0x12) even when the TRNG is operating properly (see Random number generation). This command is issued with the parameter shown in the table below.

Table 82-23. TRNG_RND Command

Register	Value	Data direction
CSE_CMD	0x14	—
CSE_P1	Random value address	Output

82.6.21 Initialize CSE

The INIT_CSE command loads the command processor firmware and memory slot data from the CSE flash memory blocks into local memory. It does not execute the secure boot protocol. The CSE firmware version is loaded into the CSE_P1 register as shown in the table below. This command must be issued before any other command when secure boot is not enabled.

Table 82-24. INIT_CSE Command

Register	Value	Data direction
CSE_CMD	0x15	—
CSE_P1	Firmware version	Output

82.6.22 Miyaguchi-Preneel (MP) Compression

The MP_COMPRESS command is a one-way compression function used to derive a 128-bit output from a given message with the parameters specified in Table 82-25. The AES ECB algorithm is used to derive the 128-bit output as described in the SHE specification. The message length input is a 64-bit value which specifies the length of the message in bits. A length error (EC=0x15) is returned if the message length is greater than 0x7fffffff (4GB).

Table 82-25. MP_COMPRESS Command

Register	Value	Data direction
CSE_CMD	0x016	
CSE_P1	Message length (bits) address	Input
CSE_P2	Message start address	Input
CSE_P3	Output address	Output

Appendix A
Release Notes

A.1 General changes

- See the changes for each individual chapter.

A.2 Introduction changes

- No substantial content changes

A.3 Memory Map changes

- Please see attached Memory map spreadsheet. Check "Revision" tab for list of changes in the memory map.

A.4 Signal Description changes

- Please see attached 'IO Signal Description and Input multiplexing tables' spreadsheet. Check "Revision History" tab for list of changes.

A.5 Clocking changes

- No substantial content changes

A.6 Reset changes

- No substantial content changes

A.7 System Boot changes

- No substantial content changes

A.8 Interrupt and DMA Overview changes

- No substantial content changes

A.9 Functional Safety changes

- No substantial content changes

A.10 Embedded memories changes

- No substantial content changes

A.11 Device Configuration Format Records changes

- No substantial content changes

A.12 SIUL2 changes

- In Table 12-1
  - Changed part from FS32R274KBK2MMM to FS32R274KSK2MMM and changed configuration from "B" to "S".
  - Changed part from FS32R274KBK2VMM to FS32R274KSK2VMM and changed configuration from "B" to "S".
  - Added "B or S" to Table 12-2.

A.13 Core Complex Overview changes

- No substantial content changes

A.14 Core description module changes

- No substantial content changes

A.15 z7 Core description changes

- No substantial content changes

A.16 Nexus changes

- No substantial content changes

A.17 z7 Nexus Module changes

- No substantial content changes

A.18 Core Debug Support changes

A.18.1 Chip-specific Core Debug Support information changes

- No substantial content changes

A.18.2 Core Debug Support changes

- No substantial content changes

A.19 z7 Core Debug Support changes

- No substantial content changes

A.20 SPE changes

- No substantial content changes

A.21 Crossbar Switch (AXBS) changes

A.21.1 Chip-specific AXBS information changes

- No substantial content changes

A.21.2 Crossbar switch module changes

- No substantial content changes

A.22 Crossbar Integrity Checker (XBIC) changes

A.22.1 Chip-specific XBIC information changes

- No substantial content changes

A.22.2 XBIC module changes

- No substantial content changes

A.23 Peripheral Bridge (AIPS-Lite) changes

A.23.1 Chip-specific AIPS-Lite information changes

- No substantial content changes

A.23.2 AIPS-Lite module changes

- No substantial content changes

A.24 System Memory Protection Unit (SMPU) changes

A.24.1 Chip-specific SMPU information changes

- No substantial content changes

A.24.2 SMPU module changes

- No substantial content changes

A.25 Semaphores2 (SEMA42) changes

A.25.1 Chip-specific SEMA42 information changes

- No substantial content changes

A.25.2 SEMA42 module changes

- No substantial content changes

A.26 Development Trigger Semaphore (DTS) changes

A.26.1 Chip-specific DTS information changes

- No substantial content changes

A.26.2 DTS changes

- No substantial content changes

A.27 Wakeup Unit (WKPU) changes

A.27.1 Chip-specific WKPU information changes

- No substantial content changes

A.27.2 WKPU module changes

- No substantial content changes

A.28 INTC module changes

- No substantial content changes

A.29 Direct Memory Access Multiplexer (DMAMUX) changes

A.29.1 Chip-specific DMAMUX information changes

- No substantial content changes

A.29.2 DMAMUX module changes

- No substantial content changes

A.30 PCM module changes

- No substantial content changes

A.31 MCB module changes

- No substantial content changes

A.32 PLLDIG changes

- No substantial content changes

A.33 CMU changes

- No substantial content changes

A.34 MC_CGM changes

- No substantial content changes

A.35 IRCOSC changes

- No substantial content changes

A.36 Enhanced Direct Memory Access (eDMA) changes

A.36.1 Chip-specific eDMA information changes

- No substantial content changes

A.36.2 eDMA module changes

- No substantial content changes

A.37 PRAMC module changes

- No substantial content changes

A.38 Flash memory controller module changes

- No substantial content changes

A.39 c55fmc module changes

- No substantial content changes

A.40 Cross-Triggering Unit (CTU) changes

A.40.1 Chip-specific CTU information changes

- No substantial content changes

A.40.2 CTU module changes

- No substantial content changes

A.41 Enhanced Motor Control Timer (eTimer) changes

A.41.1 Chip-specific eTimer information changes

- No substantial content changes

A.41.2 eTimer module changes

- No substantial content changes

A.42 Motor Control Pulse Width Modulator (FlexPWM) changes

A.42.1 Chip-specific FlexPWM information changes

- No substantial content changes

A.42.2 FlexPWM module changes

- No substantial content changes

A.43 Analog-to-Digital Converter (ADC) changes

A.43.1 Chip-specific ADC information changes

- No substantial content changes

A.43.2 ADC module changes

- No substantial content changes

A.44 Temperature Sensor changes

- No substantial content changes

A.45 Signal Processing Toolbox (SPT) changes

A.45.1 Chip-specific SPT information changes

- No substantial content changes

A.45.2 SPT changes

- No substantial content changes

A.46 Cross Triggering Engine (CTE) changes

A.46.1 Chip-specific CTE information changes

- No substantial content changes

A.46.2 CTE changes

- No substantial content changes

A.47 WGM changes

- No substantial content changes

A.48 RADAR Analog Front-End (AFE) changes

A.48.1 Chip specific AFE information changes

- No substantial content changes

A.48.2 AFE changes

- No substantial content changes

A.49 MIPICSI2 changes

A.49.1 Chip-specific MIPICSI2 information changes

- No substantial content changes

A.49.2 MIPICSI2 changes

- No substantial content changes

A.50 System Timer Module (STM) changes

A.50.1 Chip-specific STMinformation changes

- No substantial content changes

A.50.2 STM module changes

- No substantial content changes

A.51 Software Watchdog Timer (SWT) changes

A.51.1 Chip-specific SWT information changes

- No substantial content changes

A.51.2 SWT module changes

- No substantial content changes

A.52 Periodic Interrupt Timer (PIT) changes

A.52.1 Chip-specific PIT information changes

- No substantial content changes

A.52.2 PIT module changes

- No substantial content changes

A.53 CAN (FlexCAN) changes

A.53.1 Chip-specific FlexCAN information changes

- No substantial content changes

A.53.2 FlexCAN module changes

- No substantial content changes

A.54 Zipwire changes

A.54.1 Chip-specific Zipwire information changes

- No substantial content changes

A.54.2 Zipwire module changes

- No substantial content changes

A.55 SIPI module changes

- No substantial content changes

A.56 LFAST module changes

- No substantial content changes

A.57 Serial Peripheral Interface (SPI) changes

A.57.1 Chip-specific SPI information changes

- No substantial content changes

A.57.2 SPI module changes

- No substantial content changes

A.58 LINFlexD changes

A.58.1 Chip-specific LINFlexD information changes

- No substantial content changes

A.58.2 LINFlexD module changes

- No substantial content changes

A.59 I2C changes

- No changes.

A.60 FlexRay Communication Controller (FlexRay) changes

A.60.1 Chip-specific FlexRay information changes

- No substantial content changes

A.60.2 FlexRay module changes

- No substantial content changes

A.61 Ethernet MAC (ENET) changes

A.61.1 Chip-specific ENET information changes

- No substantial content changes

A.61.2 ENET module changes

- No substantial content changes

A.62 MC_RGM changes

- No substantial content changes

A.63 System Status and Configuration Module (SSCM) changes

A.63.1 Chip-specific SSCM information changes

- Added a new topic: ECC Error Monitoring.

A.63.2 SSCM module changes

- In Life Cycle Status Register (SSCM_LCSTAT) updated the footnote "Reset Value from 11 to 00".
- In ECC error monitoring
  - Removed "During flash memory scanning.....the SSCM will:
  - Updated the description to "Refer to chip-specific SSCM information section for ECC error monitoring and reaction."

A.64 Power Management Controller block (PMC) changes

A.64.1 Chip-specific PMC information changes

- No substantial content changes

A.64.2 PMC changes

- No substantial content changes

A.65 MC_PCU changes

- No substantial content changes

A.66 MC_ME changes changes

- No substantial content changes

A.67 Nexus Crossbar Multi-Master Client (NXMC) changes

A.67.1 Chip-specific NXMC information changes

- No substantial content changes

A.67.2 NXMC module changes

- No substantial content changes

A.68 Nexus Port Controller (NPC) changes

A.68.1 Chip specific NPC information changes

- No substantial content changes

A.68.2 NPC module changes

- No substantial content changes

A.69 Nexus Aurora Link (NAL) changes

A.69.1 Chip-specific NAL information changes

- No substantial content changes

A.69.2 NAL changes

- No substantial content changes

A.70 Nexus Aurora Phy changes

- No substantial content changes

A.71 JTAG Controller (JTAGC) changes

A.71.1 Chip-specific JTAGC information changes

- No substantial content changes

A.71.2 JTAGC module changes

- No substantial content changes

A.72 CJTAG changes

- No substantial content changes

A.73 JTAG Master (JTAGM) changes

A.73.1 Chip-specific JTAGM information changes

- No substantial content changes

A.73.2 JTAGM changes

- No substantial content changes

A.74 CRC module changes

A.75 Memory Error Management Unit (MEMU) changes

A.75.1 Chip-specific MEMU information changes

- No substantial content changes

A.75.2 MEMU changes

- No substantial content changes

A.76 Fault Collection and Control Unit (FCCU) changes

A.76.1 Chip-specific FCCU information changes

- No substantial content changes

A.76.2 FCCU module changes

- No substantial content changes

A.77 Self-Test Control Unit (STCU2) changes

A.77.1 Chip-specific STCU2 information changes

- No substantial content changes

A.77.2 STCU2 module changes

- No substantial content changes

A.78 Error Injection Module (EIM) changes

A.78.1 Chip-specific EIM information changes

- No substantial content changes

A.78.2 EIM module changes

- No substantial content changes

A.79 Register Protection (REG_PROT) changes

A.79.1 Chip-specific REG_PORT information changes

- No substantial content changes

A.79.2 REG_PROT module changes

- No substantial content changes

A.80 Password and Device Security Module (PASS) changes

A.80.1 Chip-specific PASS information changes

- No substantial content changes

A.80.2 PASS module changes

- No substantial content changes

A.81 Tamper Detection Module (TDM) changes

A.81.1 Chip-specific TDM information changes

- No substantial content changes

A.81.2 TDM module changes

- No substantial content changes

A.82 Cryptographic Services Engine (CSE) changes

A.82.1 Chip-specific CSE information changes

- No substantial content changes

A.82.2 CSE module changes

- No substantial content changes

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number S32R274 Reference Manual
Revision 4, 05/2018